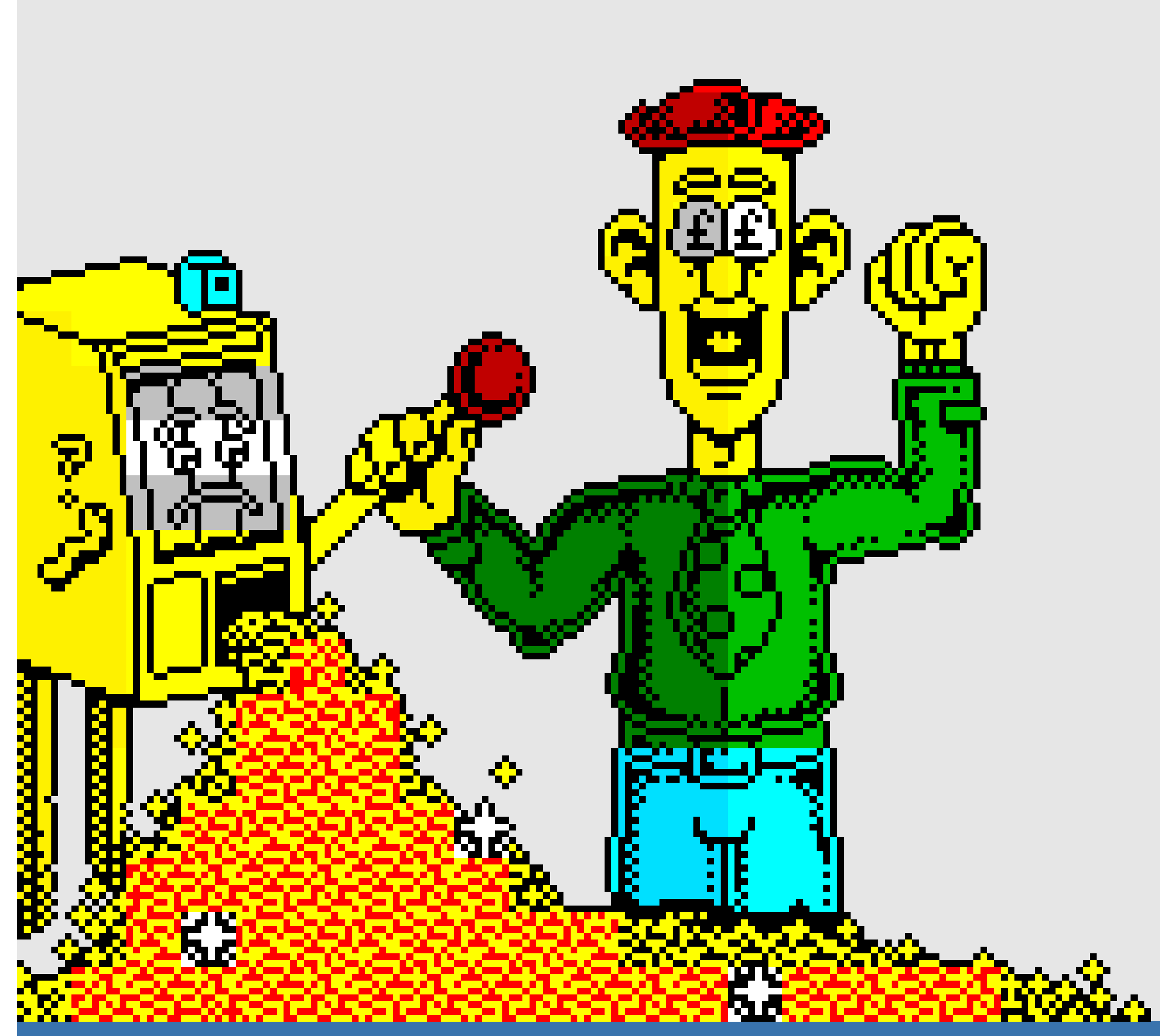
Bytemaniacos

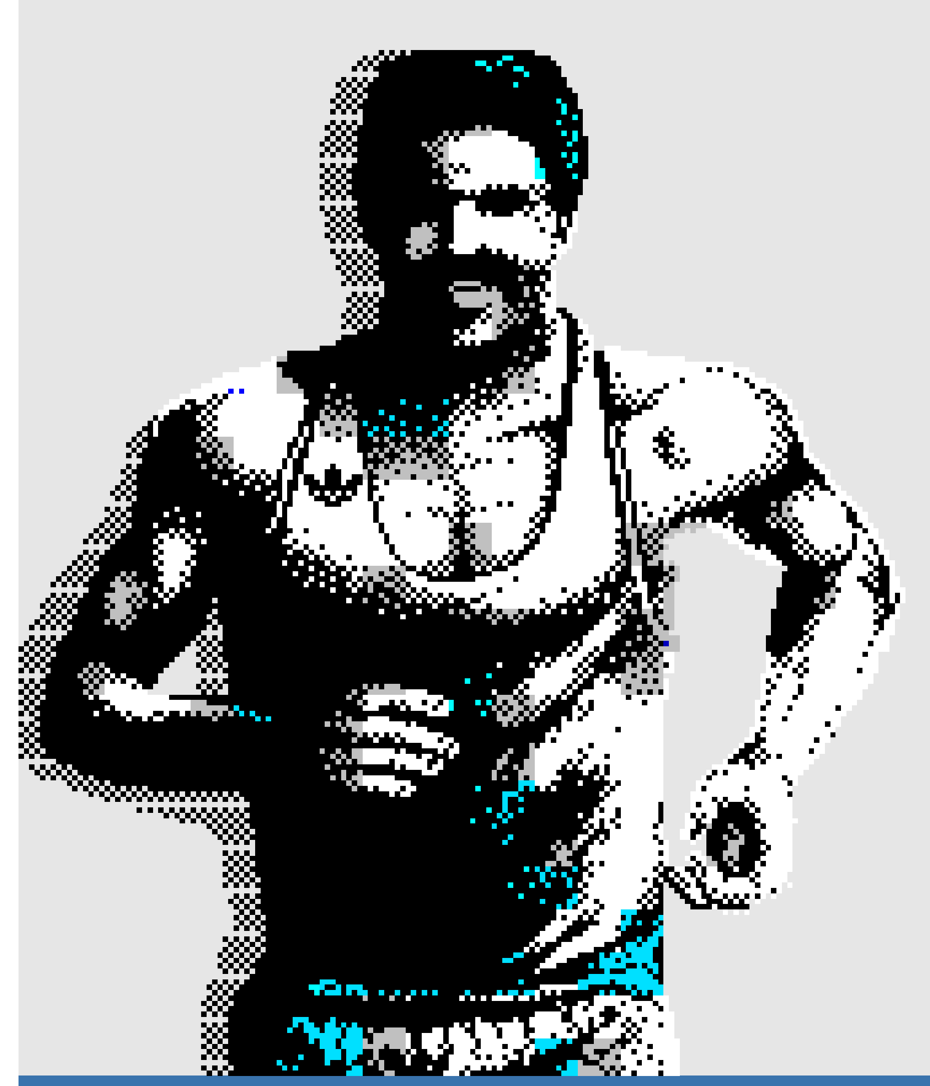
Lo más clásico en informática y videojuegos



Nº 04



“No leas porque te sientes obligado, lee porque lo que sientes es entusiasmo.”



Es absolutamente demencial lo que ha sucedido con este fanzine. 43 páginas, nuevo record, y no han sido más porque he dado el tijeretazo a los contenidos, separado el especial de Retro Madrid, y dado vacaciones a Igor (por cierto, creo que se ha marchado al caribe).

Hay mucho que contar, lo cual es muy bueno. Algunos creen que vivimos una segunda edad de oro con los 8 bits: la escena no para de sacar cosas y cada vez hay más variedad en páginas web, blogs, fanzines, etc. Creo que es precipitado lo de “segunda edad de oro”, pero desde luego es para estar contentos sin dormirnos en los laureles.

Y pasando al preámbulo de los contenidos tenemos dos grandes estrellas: Computer Emuzone y el sistema MSX. He querido rendir homenaje a ambos por distintas razones. A CEZ por su labor, dándonos alegrias contínuas en forma de nuevos juegos, y al MSX por ser el gran olvidado hasta hoy en este fanzine (4 reviews para ponernos al día).

Ya sabeis mi consejo: sofá, cervecita/refresco, y a pasar un buen rato con la lectura. Espero que os guste.

Con valor todo se puede



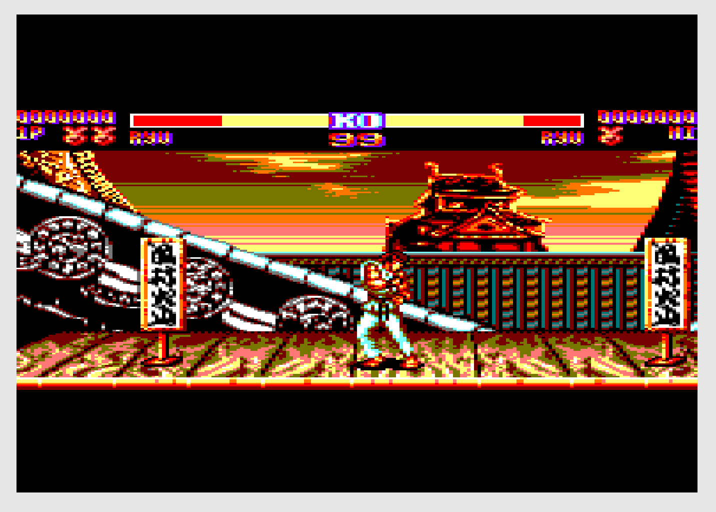
Lo que empezó como un post en un foro imaginando como sería una versión decente de Street Fighter 2 para Amstrad CPC se está convirtiendo poco a poco en una realidad tangible.

Un mallorquín, de seudónimo Dados2 está creando el proyecto, y la verdad es que cada paso dado deja mejor sabor de boca. Actualmente los decorados han avanzado bastante, al igual que los sprites, mientras que el apartado técnico sigue creciendo teniendo en cuenta las limitaciones técnicas de la máquina.

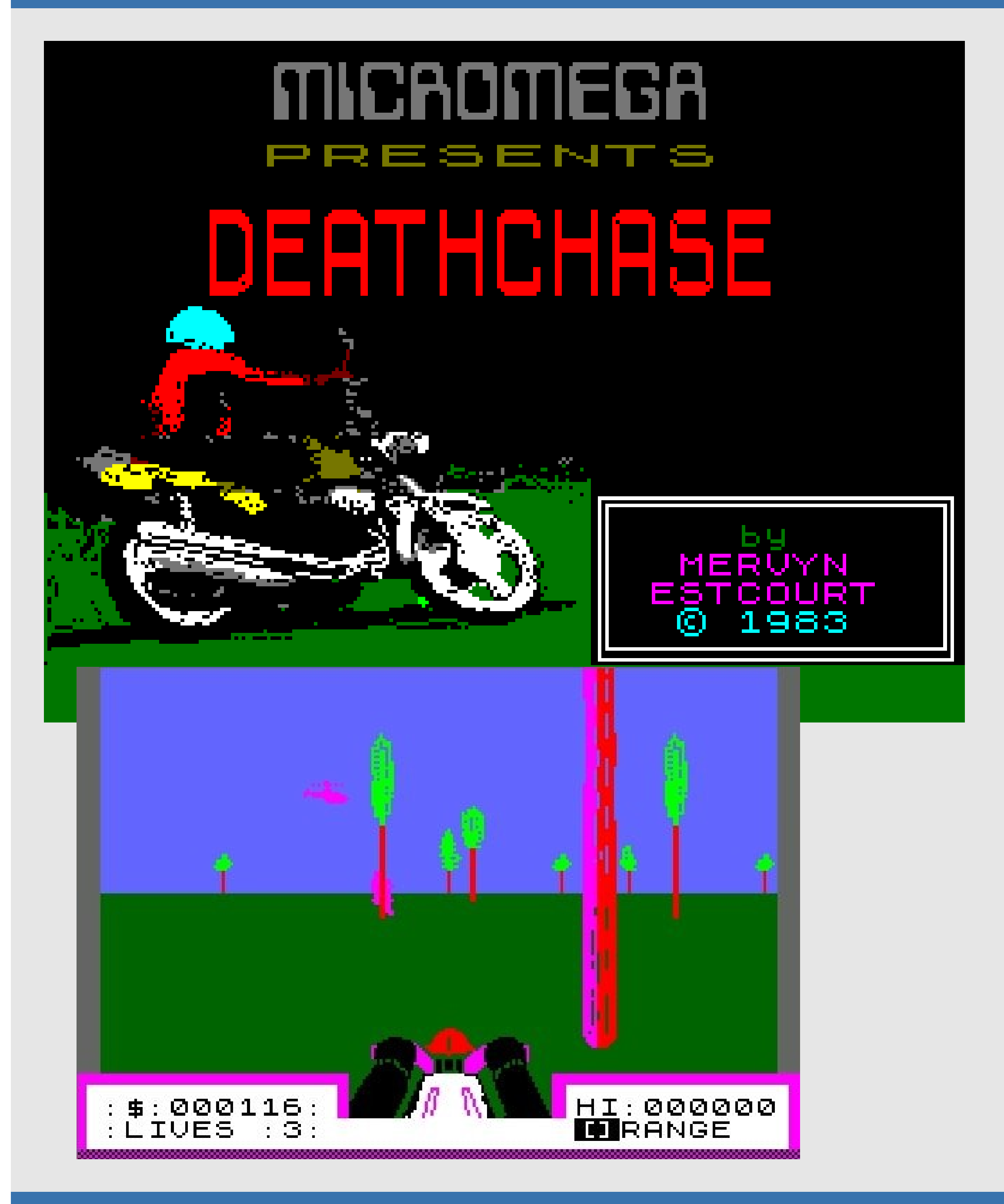
Despacito y con buena letra, lo importante es avanzar.


http://amstradcpc.mforos.com





Un ejemplo del resultado final, empleando overscan horizontal



Convirtiendo que es gerundio

Estamos en la moda de realizar conversiones a nuestro sistema favorito de casi cualquier juego que tenga algo de decencia, no es que sea algo malo pero uno piensa que es posible meter mejoras de paso.

En un tiempo record de dos semanas Richard Wilson se ha currado un port de Death Chase para Amstrad CPC. El resultado es bastante bueno en lo que a velocidad se refiere y fidelidad respecto al original, pero no aprovecha lo que puede llegar a hacer un CPC gráficamente. Comprendemos que es decisión del autor, pero da lástima perder una oportunidad así.

Entre las leves mejoras tenemos una corrección del efecto de mezcla de colores que había en ZX Spectrum, aparte de una pantalla de presentación levemente retocada y que presenta más de 4 colores en modo 1.

Con lo que molaría una opción en el menú que cambie los gráficos por cierta acción en el planeta de Endor...

http://bitwise-systems.com/files/deathchase.zip

http://www.cpczone.net/boards/viewforum.php?f=4

Otro SUdoku para Sinclair QL

¡Ya era hora!

Desde los foros de www.speccy.org hemos asistido a un parto con mucho retraso: el primer juego del siglo 21 para Sinclair QL. Así dicho parece fuerte, pero es realidad, la escena de QL parece que empieza a moverse no sólo en el terreno de aplicaciones.



El juego elegido es un Sudoku, muy apropiado para un ordenador pensado para trabajar, y está elaborado en lenguaje Super BASIC. Su desarrollo todavía está en curso, con mejoras gráficas y de opciones, pero es jugable en su actual forma.

Como pega tiene que usa toda la pantalla, por lo que si usas un QL con una TV tendrás problemas.

Esperemos que sea el inicio de una nueva época de juegos para este sistema.

http://www.speccy.org/foro/viewtopic.php?f=15&t=61



GTW 64! sigue de pesca

En esta ocasión los persistentes componentes de GTW 64! han conseguido encontrar el juego Rockus.

Se trata de un juego que quedó en segundo puesto en un concurso de los años 90, lo que hizo que no se le prestase mucha atención. Pero con el tiempo siempre queda la incógnita de cuan bueno era, y tras mucho buscar aquí lo tenemos.

Se trata de un juego de plataformas de avance horizontal, algo como Super Mario Bross pero sin scroll (me recuerda poderosamente a mi Fantastic Adventure).

Tiene el mérito de estar realizado en su mayoría con BASIC, dejando para el ensamblador las tareas más costosas (ej. gráficos). El resultado es agradable, pero adolece de tener unos enemigos que aparecen siempre en el mismo punto de la pantalla, lo que deja poca estrategia a la hora de avanzar y facilitando el "corre todo lo que puedas".

http://gtw64.retro-net.de/Pages/r/Review_Rockus.php

Por cierto, siguen tras la pista de Darius+.



La moda de los puzzles llega a Atari XE/XL

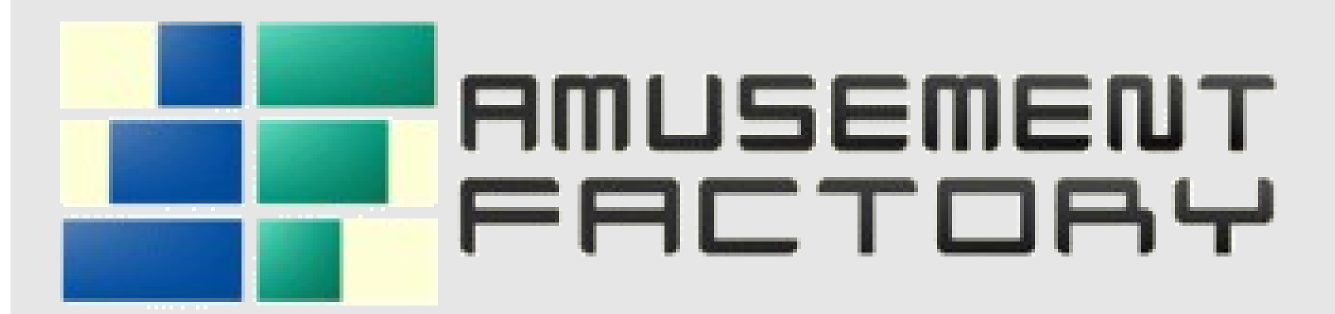
De forma conjunta a Flop Magazine, una publicación checa dedicada a Atari, se ha lanzado el juego Swapz.

La verdad es que es una grata sorpresa, ya que por todos es conocido este puzzle (unir piezas en vertical u horizontal en filas de 3 o más), pero no exisitía una versión para los 8 bits de atari.

Lo bueno es que el sonido se ha cuidado bastante, desde la música del menú a los sonidos de las piezas, y los gráficos son bastante agradables.

Para rematar el juego se han añadido algunos efectos en el menú de opciones y en la tabla de clasificación, lo que al menos ayuda a que no quede sosilla la cosa a pesar de ser un puzzle.

http://atari.fandal.cz/detail.php?files_id=5770



Una factoría de conversiones

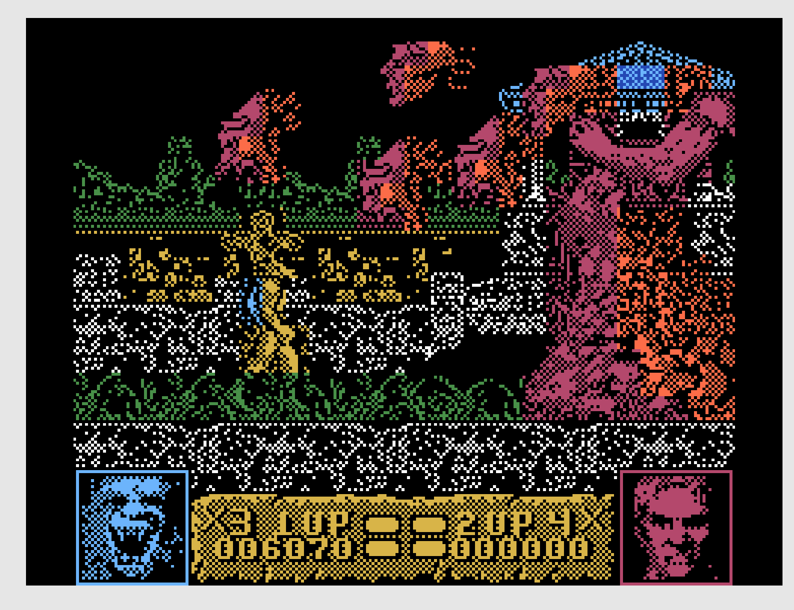
Es curioso encontrar un grupo de programación para MSX dedicado casi en exclusiva a realizar conversiones de juegos para su sistema desde otras plataformas, y a mejorar incluso juegos existentes.

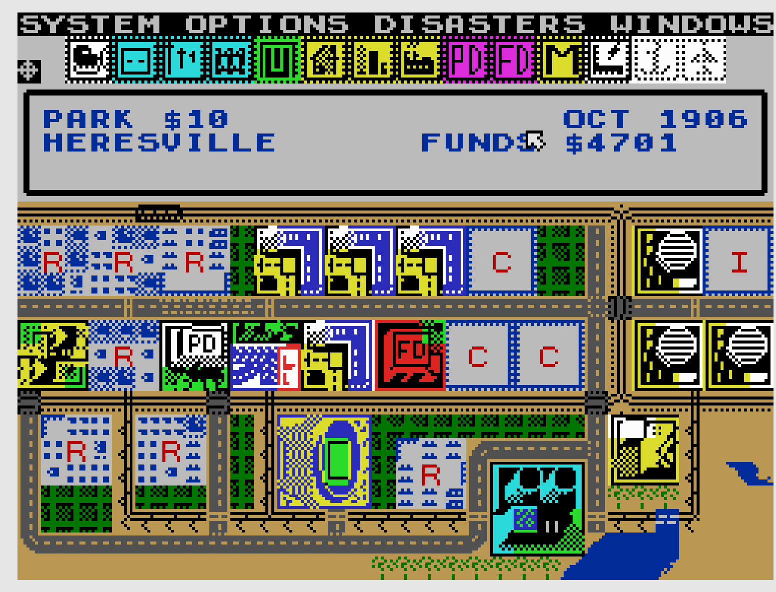
Knightmare Gold es una versión mejorada del clásico de Konami para MSX2, con mejora de la paleta de colores, sprites, y algún que otro detalle. El resultado no está nada mal, pero siempre que tengas un MSX con una CPU rápida (olvidad los 3.57 Mhz si quereis scroll suave).

Respecto a conversiones hay de todo: Altered Beast, Sim City, Ghost'n Goblins, etc. Todo conversiones desde ZX Spectrum que lo único que aprovechan es una mejor paleta de colores pero no usan los mejores gráficos de MSX en su mayor parte. Un ejemplo es Ghost'n Goblins, con un mapa genial, música sacada Amstrad CPC y de Zx Spectrum 128K (este último del Ghouls 'n' Ghost), pero que tiene un desarrollo con gráficos de ZX Spectrum que desmerece el conjunto.

Visitadlos porque pasareis un buen rato.

http://www.caetano.eng.br/main/index.php?l=en&menu=software









Especial

Computer Emuzone

http://computeremuzone.com/

Son gente de muchas partes, especialistas en algo concreto o un poco de todo, pero con una misma afición y un mismo objetivo: realizar juegos para ordenadores de antaño o remakes para PC's de hoy día.

Son más de una treintena de programadores, grafistas, músicos, y testeadores, todos agrupados en un mismo foro y trabajando codo con codo para proporcionar juegos de calidad a la escena. Un equipo con una creatividad, pasión, y dedicación sólo comparable a las grandes compañías de videojuegos de los años 80.

Os invito conocer de primer mano lo que se cuece en esta factoría de ilusiones, lo más cercano a la Light & Magic desde el punto de vista de la retro informática.

Especial Computer Emuzone

Remake Ranarama

Salto a salto se completa un gran proyecto

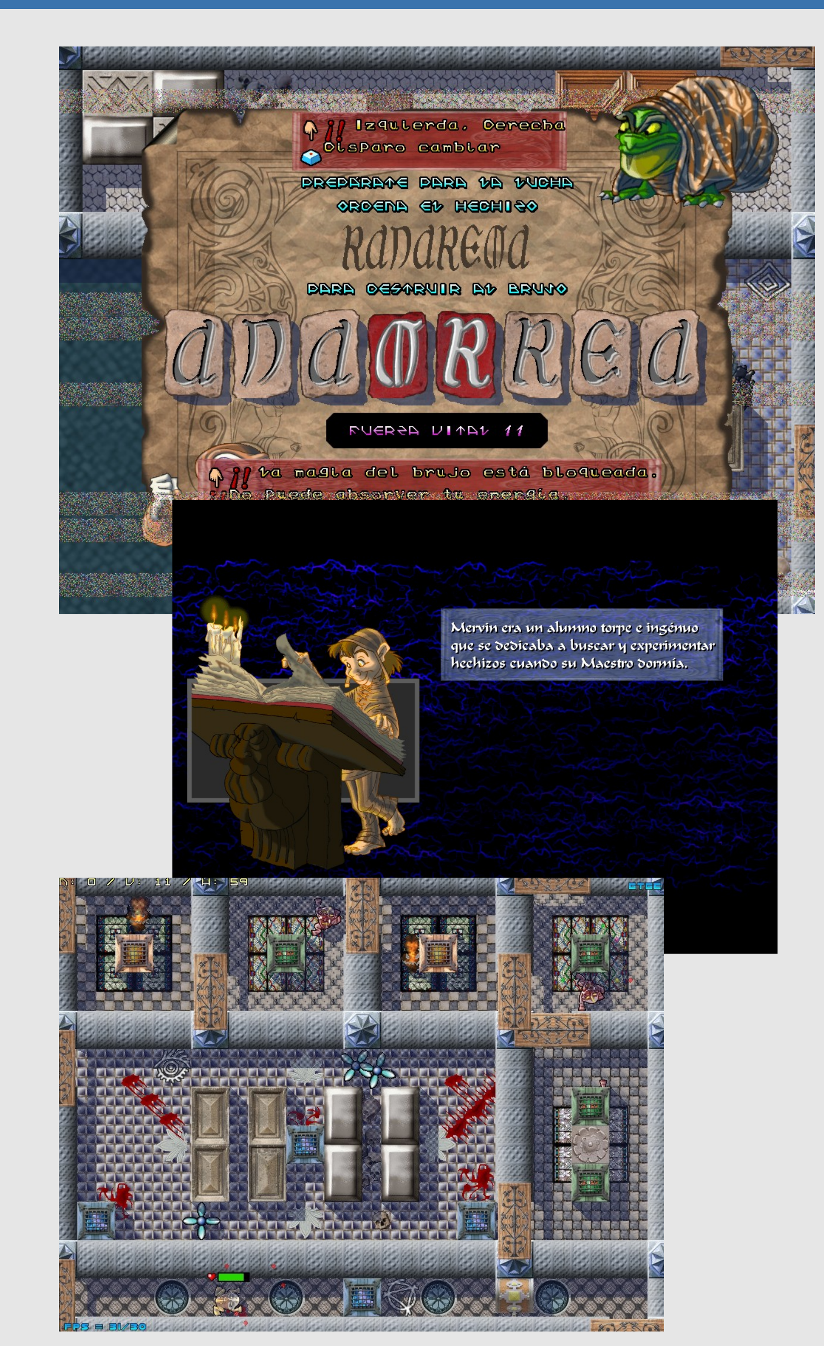
Por muchos de los lectores es conocido el juego Ranarama de Hewson Consultants, en el cual llevábamos a un joven aprendiz de mago convertido en rana debido a su torpeza y enfrentado a toda una legión de bichos y magos.

Muchos lo consideran un clon de Gauntlet, pero nada más lejos de la realidad. Lo original de este juego era la forma de combatir, mediante un puzzle en el que teníamos que ordenar la palabra “Ranarama”, un sistema algo confuso al principio pero que enganchaba con ganas si se le daba una oportunidad.

CEZ se está esmerando mucho en este remake, una de sus grandes bazas junto a Capitán Sevilla, y presenta unos gráficos realmente elaborados, junto a imágenes de introducción de altísima calidad.

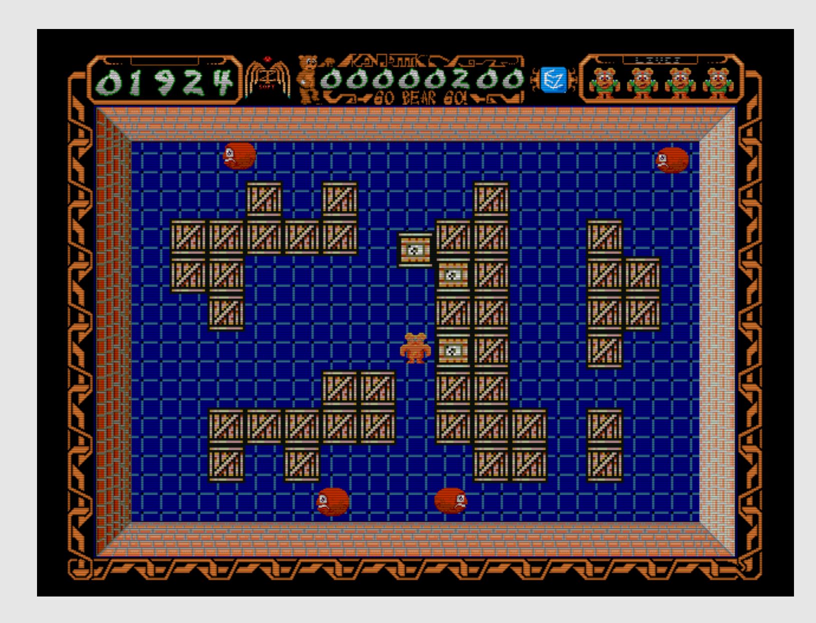
Otra de las novedades va a ser el final, totalmente distinto al juego original y que promete ser una gran sorpresa.

Todavía queda un poco para disfrutar de las aventuras de Marvyn, pero la espera se va a ver recompensada con creces.



Especial Computer Emuzone

Remake Go Bear go!



Un clon de Pengo muy divertido

Go Bear go! es un juego desconocido para la mayoría.

Apareción originalmente en el número 73 de la revista Sinclair User, como un juego de los que se incluyen de regalo con la revista, y para ser un clon del conocido Pengo resultaba realmente bueno.

Nuestro objetivo no puede se rmás sencillo, destruir todos los enemigos a base de empujar las cajas del escenario. Evidentemente no todos los niveles serán tan sencillos, habrá cajas especiales, enemigos más listos, menos opciones para empujar, etc.

Mientras escribía este fanzine ya ha salido a la luz:

http://cezgs.computeremuzone.com/ficha.php?id=16

También podeis probar el original en:

http://www.worldofspectrum.org/infoseekid.cgi?id=0002067



Especial Computer Emuzone

Remake Turbo Girl

Para mi la titi, para ti la moto

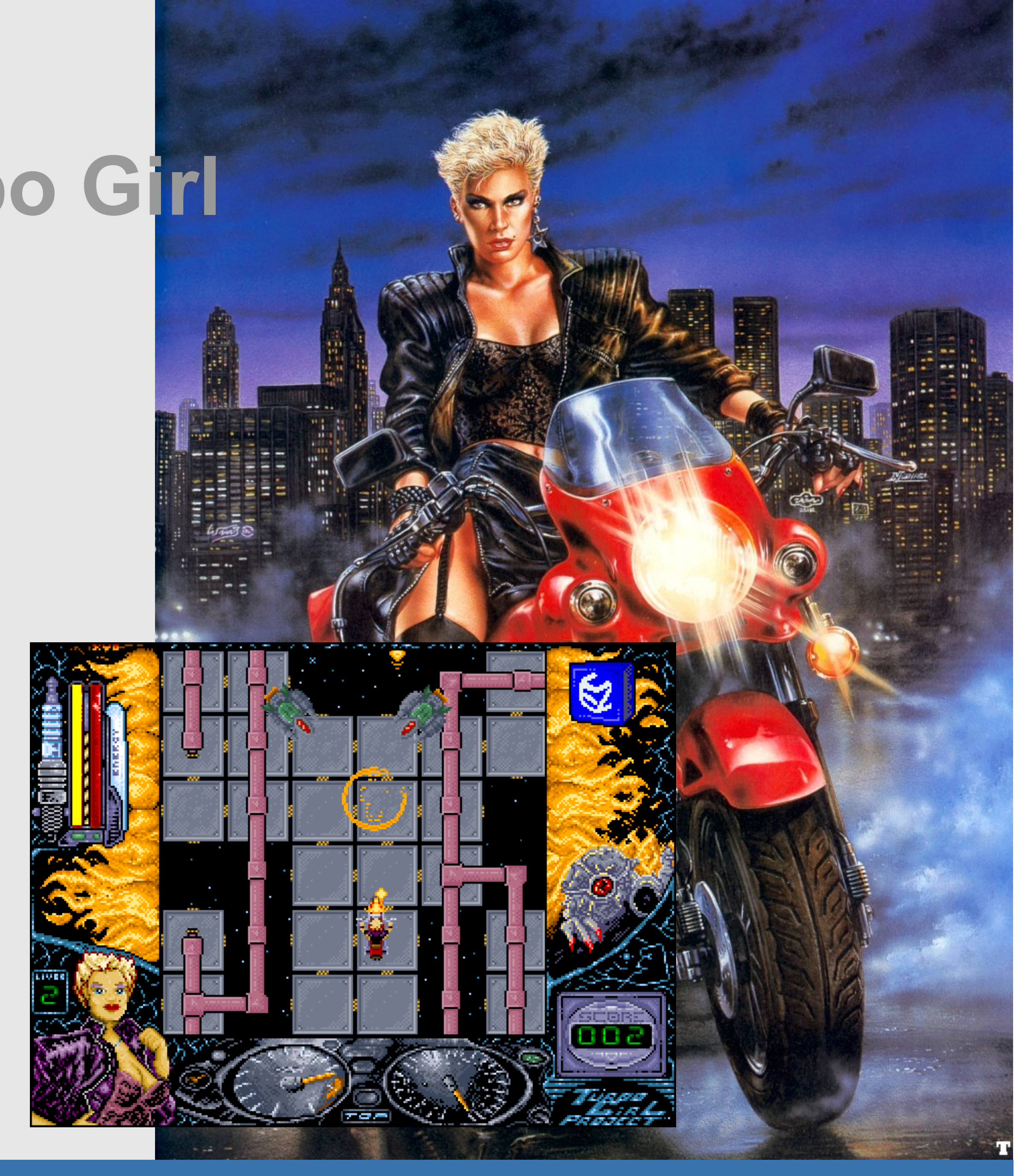
En esta ocasión no hay mucho que comentar, llevamos a la señorita inocente de la derecha, ropa interi... digo, moto incluida, en un juego de acción tipo matamarcianos con scroll vertical.

La gracia está en que debemos mantener la moto sobre el suelo, sin caer al vacío, por si no tuviéramos bastante con mantenernos concentrados en los pech... digo en los enemigos que nos atacan.

No sabemos si el juego tendrá algo más de picaresca que el juego original (molaría una pose entre fase y fase de la protagonista) o un final a la altura que mejore el original (una teta o algo así).

Eso si, sólo con la portada de Luis Rollo, y que hagan un poco más jugable el jueguecito, ya nos tienen encandilados para una buena temporada.

Esperamos el resultado con ilusión.



Especial Computer Emuzone

Remake Capitán Sevilla



El proyecto más esperado

Si tenemos que destacar un proyecto como "estrella" es este. No sólo porque se trata de un juego magnífico de por si, sino por el trabajo tan detallado que se está realizando en su adaptación a los nuevos tiempos.

Gráficamente está a la altura del mítico Sir Fred, remake que ha traspasado fronteras por su calidad, y tras ver la demo en la pasada Retro Madrid uno cuenta cada día del calendario.

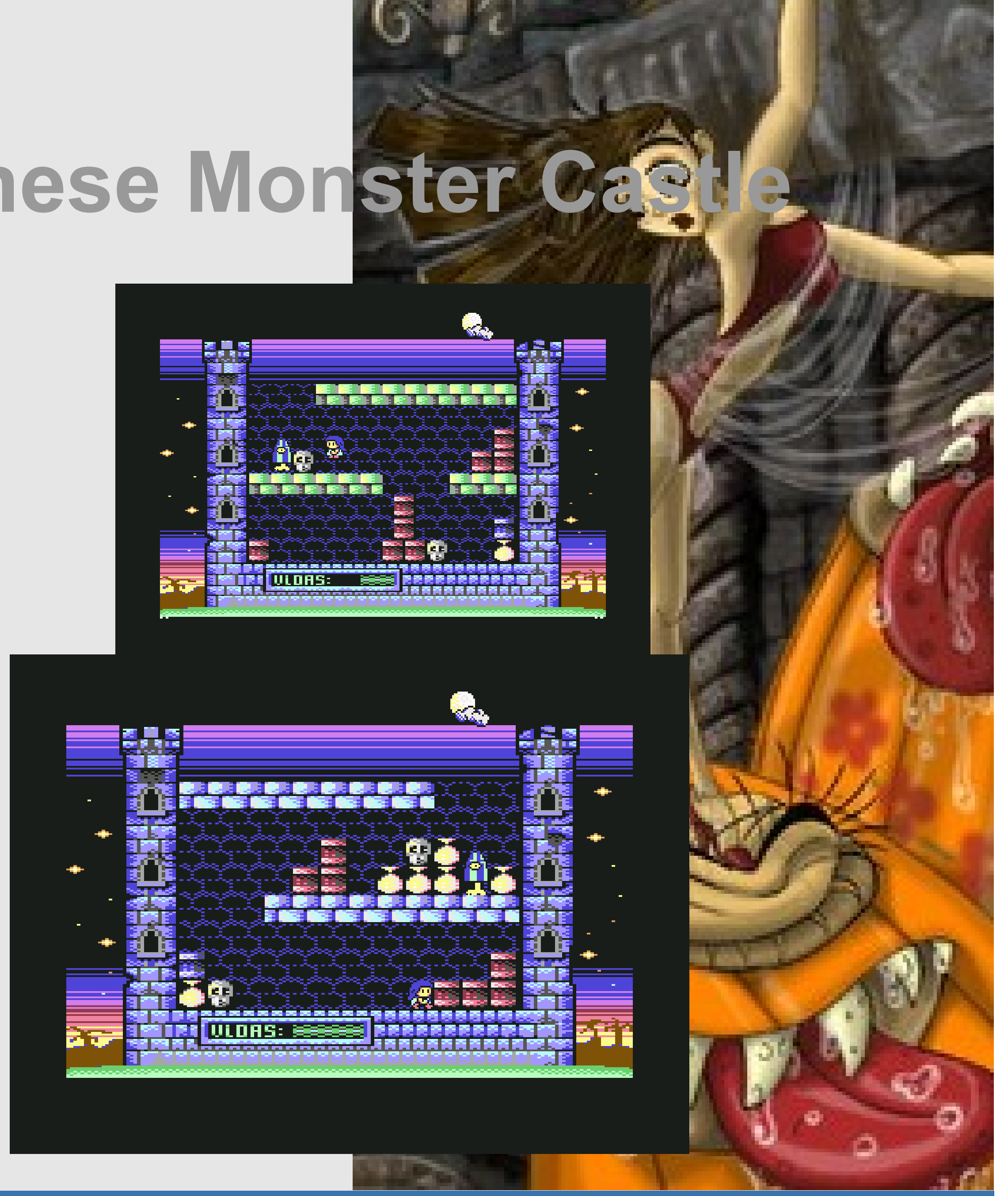


En movimiento gana mucho, aún más que las geniales imágenes que podeis ver, y todo apunta a éxito seguro.

La primera parte ya está terminada, recordemos que se compone de dos, y es muy posible que este mismo año pueda ver la luz.

Especial Computer Emuzone

C64 Classic Japanese Monster Castle



Commodore existe en España

Pasamos a los proyectos para ordenadores de 8 bits, y no hay mejor comienzo que presentar el primer juego de CEZ para los Commodore 64.

Los usuarios de ZX Spectrum ya saben de las bondades de este juego, un buen estruja cerebros que proporciona horas de diversión. El objetivo es llegar a la parte superior de la pantalla usando los objetos que hay repartidos, una especie de puzzle plataformero.

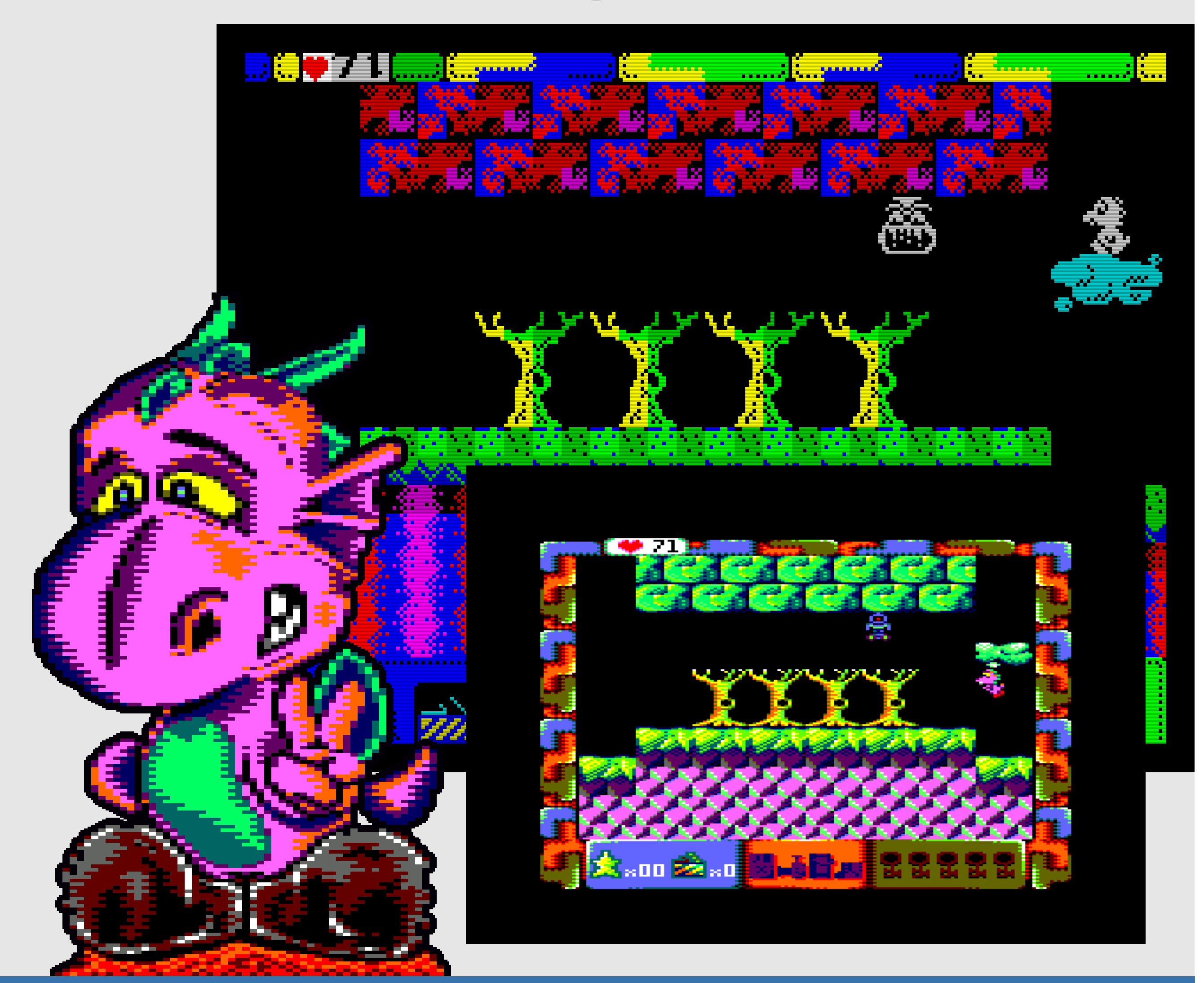
Todo el aspecto técnico se está adaptando a la nueva máquina de la mejor forma posible, con gráficos que aprovechan bien las capcidades del C64 y unas melodías mejoradas que aprovechen las posibilidades del chip SID.

Esperamos que no sea el único juego que aparezca en esta plataforma, y que sea el comienzo de una buena relación con los usuarios de C64.

No queda mucho, anda bastante avanzado.

Especial Computer Emuzone

ZX/CPC Mariano the Dragon



En ocasiones veo dragones

Os presentamos a Mariano, un dragón muy peculiar. Visitará a los usuarios de ZX Spectrum y Amstrad CPC en un juego plataformero con un mapeado bastante extenso.

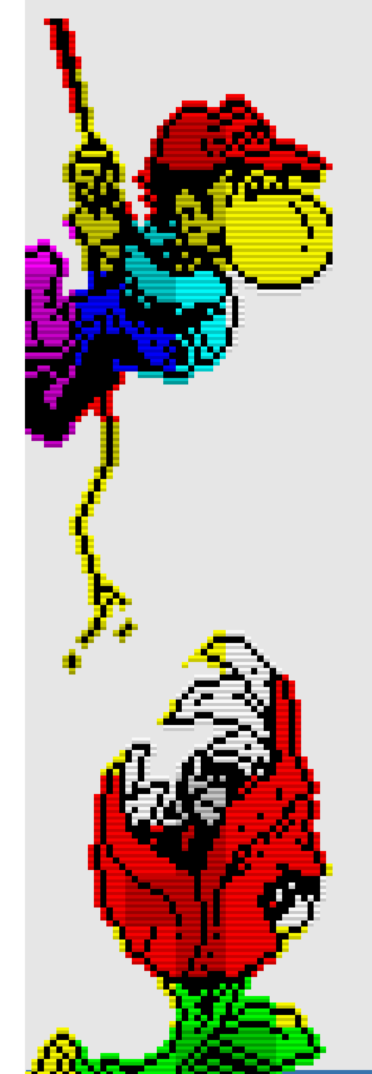
La demo de Retro Madrid andaba algo verde en cuanto a desarrollo, le faltaba algo de consistencia, pero precisamente se mostraba para recabar la opinión de los usuarios.

El juego está avanzado, falta por pulir un poco la velocidad, el movimiento de los enemigos, y darle un poco de salsa al asunto, pero las bases son muy prometedoras.

Desde aquí mucho ánimo a los autores, estamos seguros que saldrá algo grande con un poco de empeño.

Especial Computer Emuzone

ZX Spectrum JINJ



Una grata sorpresa

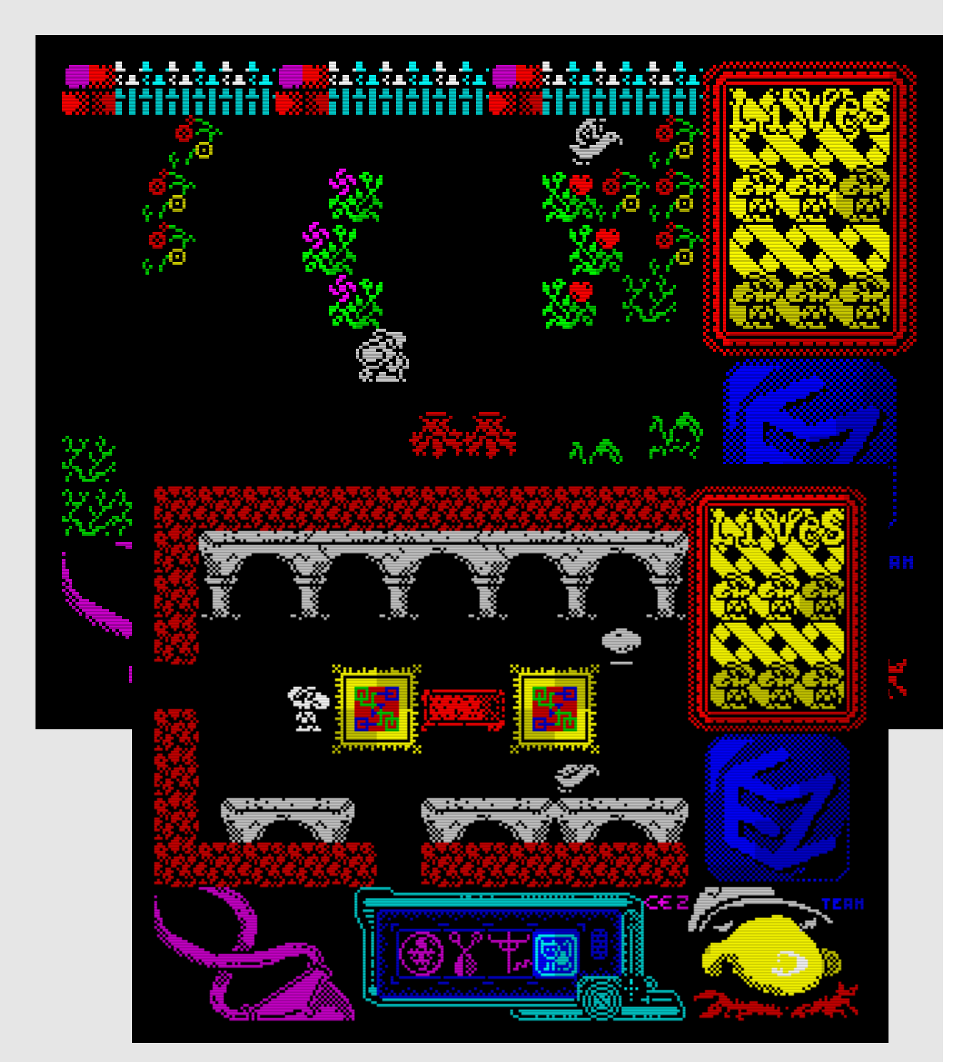
JINJ es un proyecto prácticamente desconocido, pudimos ver un adelanto en Retro Madrid y no pudimos quedar más complacidos.

El desarrollo recuerda poderosamente al clásico Saimazoom de Dinamic, pero en este caso todo desprende encanto y personalidad.

Una variedad rica de escenarios, gráficos con garra, un menú de opciones muy vistoso y colorido... todo está ensamblado con mimo y cuidado al detalle.

El lanzamiento ha sido hace muy poco, y en el próximo número daremos buena cuenta de este juego en una review.

http://cezgs.computeremuzone.com/ficha.php?id=



Especial Computer Emuzone

ZX Spectrum Box Reloaded

Proyecto secreto

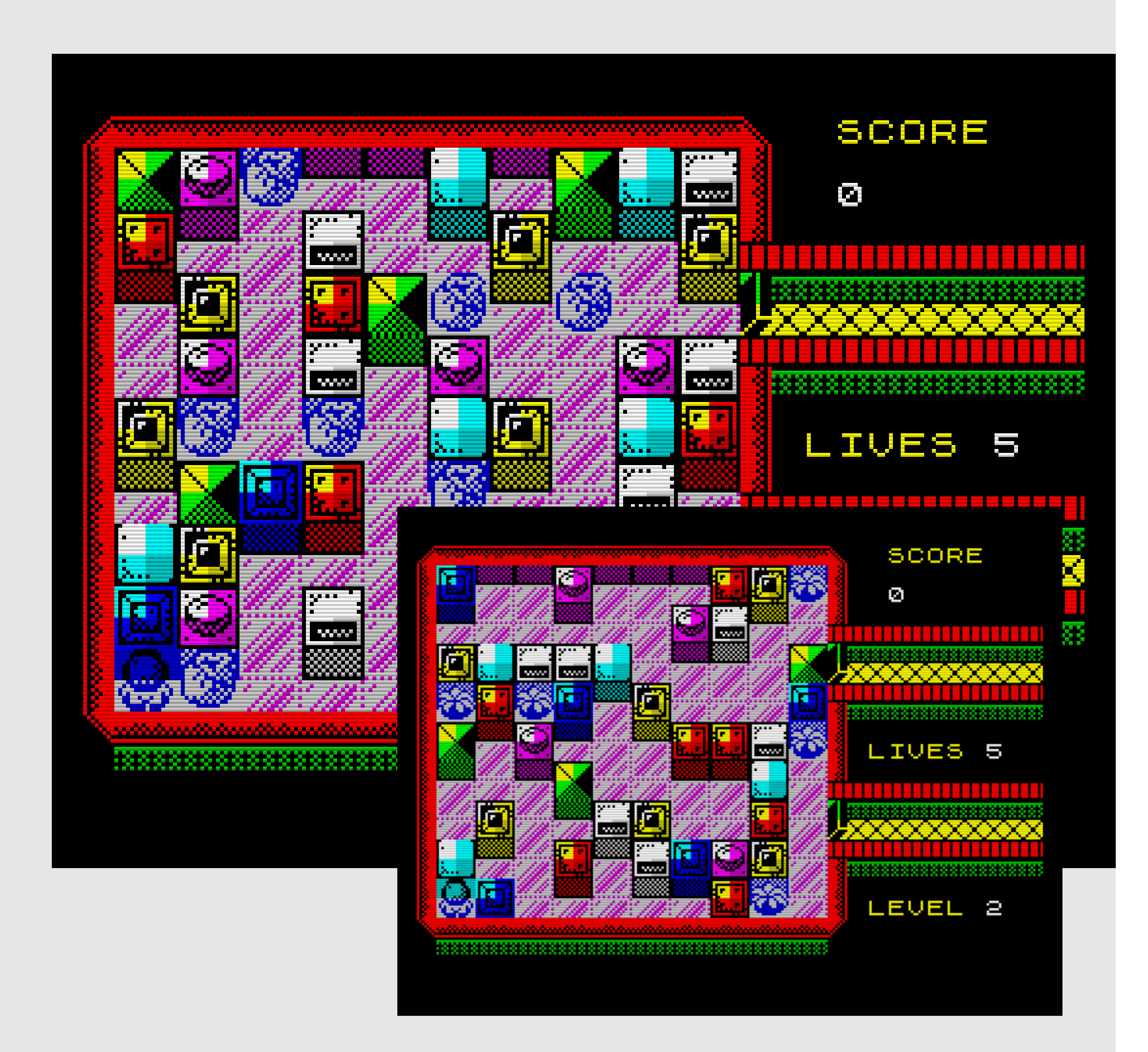
Poco, muy poco podemos desvelar de Box Reloaded, aún cuando su desarrollo se encuentra muy avanzado.

Se trata de un juego muy parecido a Sokoban, hay que recoger los objetos dispergados por la pantalla a base de mover cajas, y poco más.

El aspecto gráfico es inmejorable, el color inunda la pantalla pero de forma ordenada y vistosa, mientras que los sprites se han cuidado para tener todo tipo de detalles.

No hay más información sobre él, ni fecha de salida ni más detalles sobre el juego, así que quedamos en las manos de su autor para saber más sobre esta interesante producción.

Que falte poco!



Especial Computer Emuzone

ZX Spectrum Ilogic All

Afilad los lápices

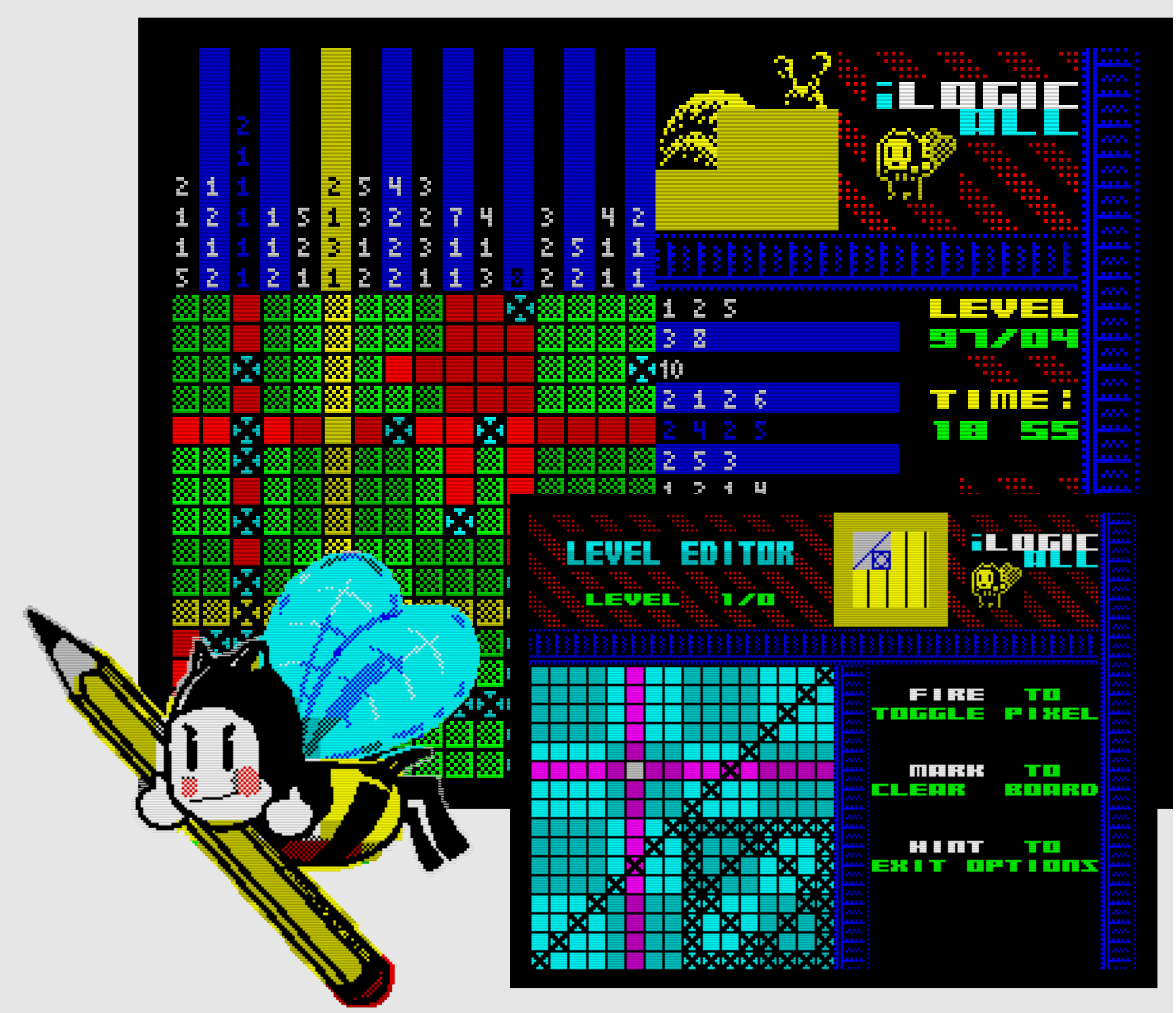
Y aquí tenemos de nuevo a los Mojon Twins, con una producción basada en el conocido Picross.

Nuestro objetivo es basarnos en los números de los laterales para saber cuantas casillas deben estar coloreadas en cada fila, así de simple... y de jodido.

Lo que aquí veis es una mezcla de buscaminas y sudoku pero con muy mala leche, y en la que el indicador de tiempo límite conseguirá sacar de quicio a más de uno. Ya vereis, ya.

El juego va a ir acompañado de un editor, para poder fastidiar a familiares y conocidos con niveles más demenciales. Una herramienta imprescindible para masoquistas y amigos de lo imposible.

Quedais avisados, soñareis con abejas zumbando alrededor de vustra cabeza...



Especial Computer Emuzone

ZX Spectrum I need speed

Quemando goma

Y acabamos este repaso a los proyectos de CEZ con uno de los más innovadores. Si, señores, no es un puzzle ni un juego de plataformas, es un juego de carreras de fórmula 1.

El aspecto es muy parecido a F1 Spirit de MSX, y aunque muchos opinen que es un clon nada más lejos de la realidad (el autor quedó sorprendido al saber de dicho juego, no lo conocía).

Aunque el juego permite dos jugadores, a pantalla partida, y cada uno con su scroll, sorprende lo rápido que se mueve todo.

Nos falta saber si habrá boxes y otros detalles que complementen la experiencia, pero de momento ya nos sentimos muy esperanzados por ver un juego que se aparte un poco de los demás.

Felicidades por anticipado al autor.



Especial Computer Emuzone

Despedida pero sin cierre



Y llegamos al final

No hemos puesto todos los que son, hay más cosas escondidas en las entrañas de CEZ. Os podemos hablar de los geniales artworks que hemos atisbado de Zombie Calabera, del prometedor remake de Fernando Martín,del elaborado Uwol que esperamos una pronta aparición, y así un largo etcétera.

Lo importante es que hay mucho movimiento, ideas, y proyectos, lo cual nos hace sentirnos muy alegres por gozar de una escena tan viva y rica en nuestro país.

Pero no sólo son proyectos toda su labor, poco a poco se están volcando con llevar a los usuarios copias físicas de calidad de los últimos lanzamientos, incluso algún recopilatorio, que no sólo vamos a poder adquirir en las reuniones o ferias. Ya queda menos para esa tienda on-line.

Y es que tan importante como tener una escena viva lo es el que exista una opción profesional a la hora de tener las novedades. De no ser así no podríamos disfrutar de nuevos cartuchos de MSX, discos de 3” en ZX Spectrum y Amstrad CPC, etc.

Desde estas líneas, gracias.

Elige tu propia aventura

Rescatamos un concurso del olvido

Hace años los libros de "elige tu propia aventura" eran muy conocidos. En cada página nos narraba la historia y luego nos daba opciones el texto a saltar a otra página según nuestra elección.

Requisitos

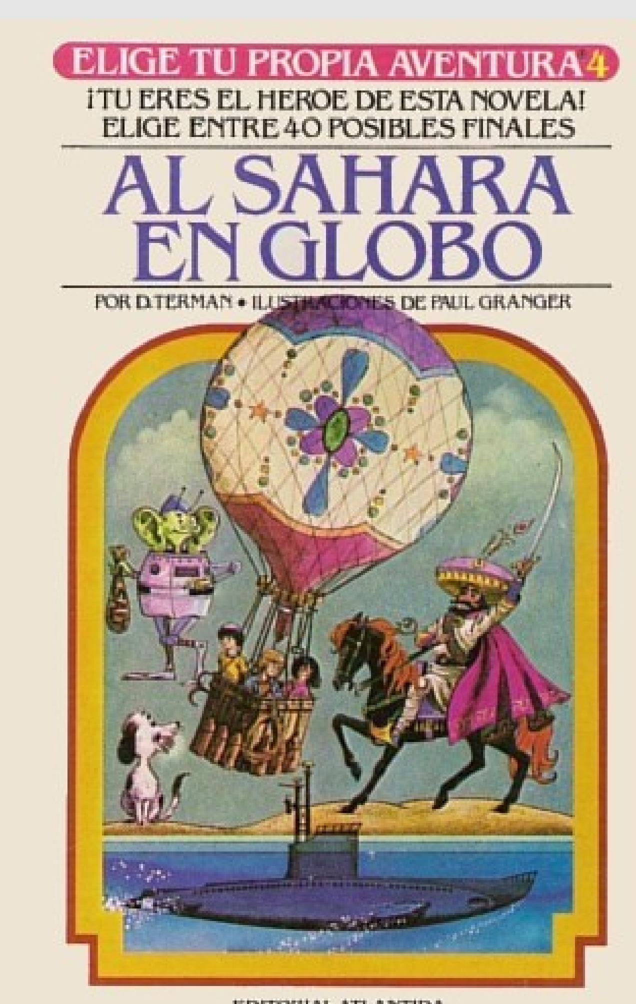
Se admite cualquier lenguaje de programación, basta con que funcione en un ZX Spectrum 48K. Como la ocasión lo merece se admite el uso de micro Speech, pero no debe ser obligatorio su uso, sólo opcional. Si hay gráficos sólo se admiten caracteres estándar, ni tan siquiera GDU's. Se pueden ver ejemplos de cómo usar los gráficos en:

http://computeremuzone.com/forum/viewtopic.php?t=1196

Se pueden usar objetos, pero siempre dentro de las opciones de elección.

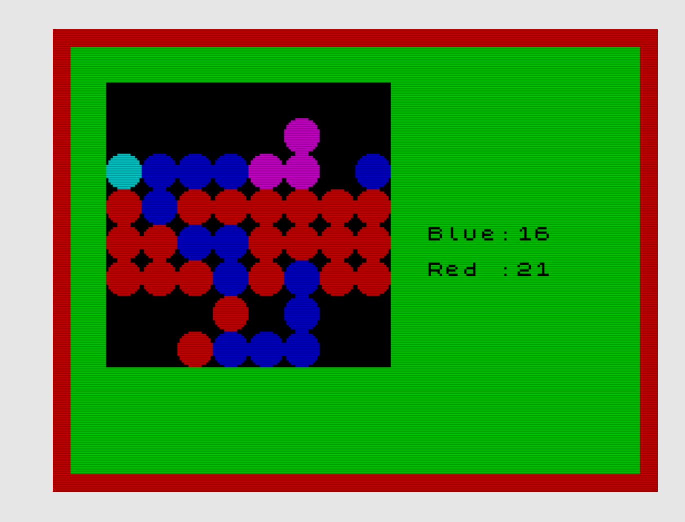
Forma de presentación y plazo

Mandar un e-mail a kurdoman@gmail.com, a ser posible uno de ellos en modo texto por si no llegan adjuntos. La fecha tope es el 23 de Mayo de 2008 inclusive.



Minigame Compo 2008

http://minigamecomp.org.uk/index.htm



Realizar juegos en 1K puede parecer algo de ciencia ficción hoy día, pero es perfectamente posible. En esta competición se pueden presentar juegos en categorías de 1, 2 y 4K, para cualquier tipo de ordenador/videoconsola de 8 bits.

Desde un Othello en un miserable K, a juegos de plataformas con gráficos más que decentes en cuatro. Todo es posible en esta competición.

Si quereis participar las fechas límite son:

1K – 31 de Julio de 2008
2K – 30 de Septiembre de 2008
4K – 30 de Noviembre de 2008

El formato debe ser el más común en emuladores de cada sistema.

Ahora mismo no hay muchos juegos presentados, pero si es posible ver y descargar juegos del año pasado, lo cual es muy recomendable porque se alcanzó un buen nivel.

Suerte a los participantes.





Un juego ganador

CAOS Begins no es sólo un buen juego de MSX, es también el ganador del concurso MSXDev del año pasado, lo cual es garantía de calidad debido al nivel de los participantes en las últimas ediciones.

El juego bebe de las bases sentadas por Maze of Galious (aunque nos recuerda más a Treasure of USAS en cierta forma): mazmorras que debemos explorar castillo tras castillo.

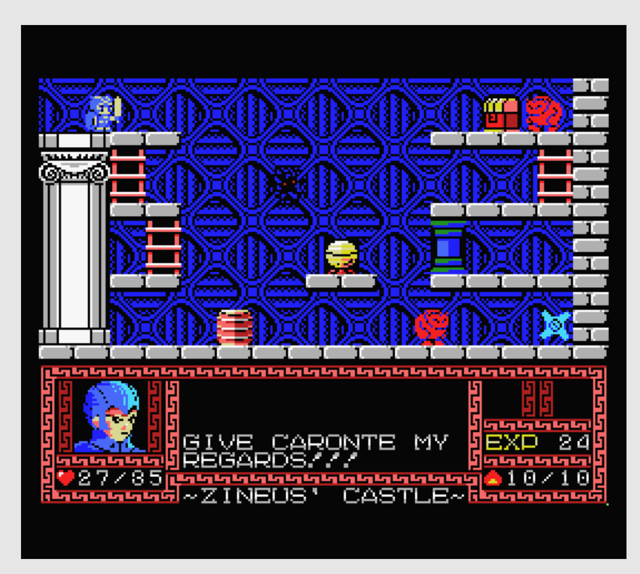
La historia nos situa como el rey Zineus, el cual va en busca de su compañera de fatigas Aleena y de paso pretende ganarse el favor de Zeus. Básicamente es eso, con una historia que es mejor leer.

Gráficos

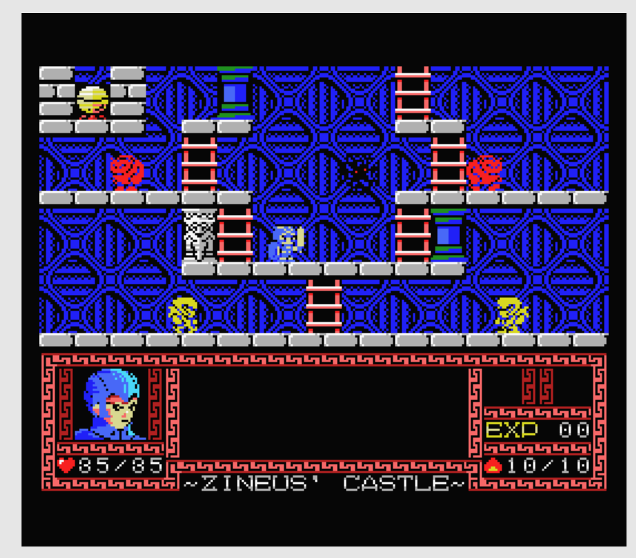
El trabajo es realmente impresionante, estamos hablando de un diseño para MSX de primera generación que no tiene nada que envidiar a los títulos de los años 80 en formato cartucho.

El protagonista es perfecto, mientras que los enemigos son a un color debido a las limitaciones de la máquina.

Pero el Do de pecho está dado en los escenarios, muy detallados y ricos en variedad a lo largo de los castillos.



El scroll está muy conseguido y es bastante suave



La paleta de colores es muy agradable, mejorando todo

Sonido

Hay varias melodias, tanto en la presentación como en el transcurso del juego, y todas ellas son de un acabado profesional. Muy amenas y ricas en matices.

Los FX no desentonan con la música, y combinan a la perfección.

Divertido y adictivo

Pero tener un nivel técnico excelente no significa que el juego también lo sea, y en este caso nos vemos con un juego que vale cada euro del cartucho.

Es muy divertido, y su gran baza es que los mapas no son tan grandes como para perderse, por lo que una vez aprendidos nos es más sencillo avanzar e incitan a pasar de nivel. Eso si, hay que armarse de paciencia.

El precio no es problema, sólo 18 euros e incluye los portes (o eso parece), por lo que no hay excusa.

Gráficos	9
Audio	9
Juego	9

Nota 9

Lo podeis adquirir por Internet en:

http://www.msxcartridgeshop.com/

También disponible en ferias y RU's.

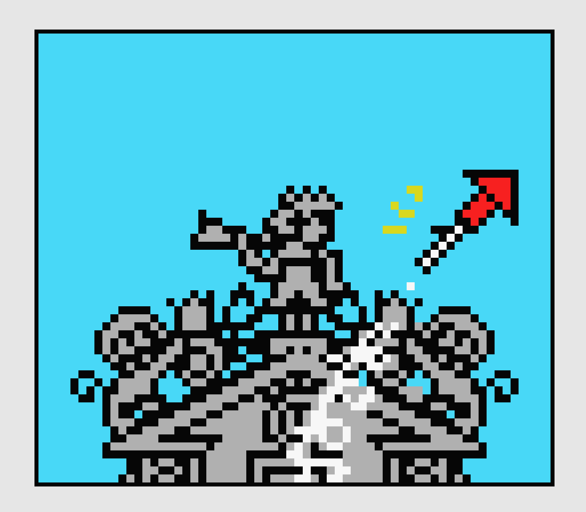
También está disponible para emular:

http://msxdev.msxblue.com/index2.htm



Txupinazo!





Gráficos 8
Audio 9
Juego 7

Nota 8

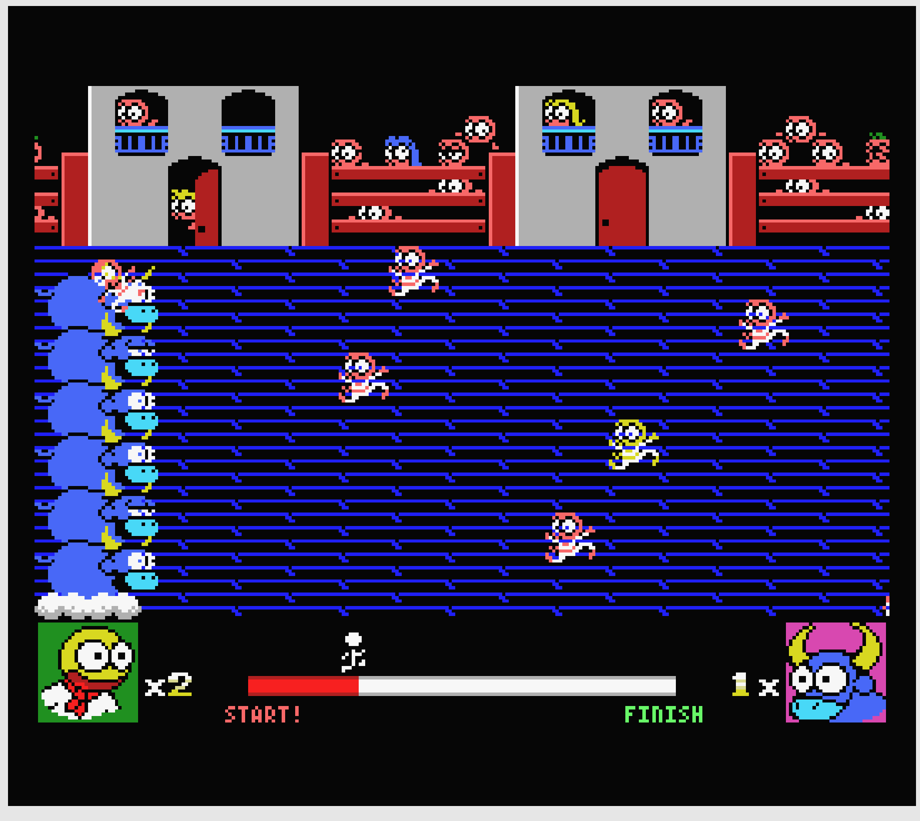
La mejor forma de correr los San Fermines

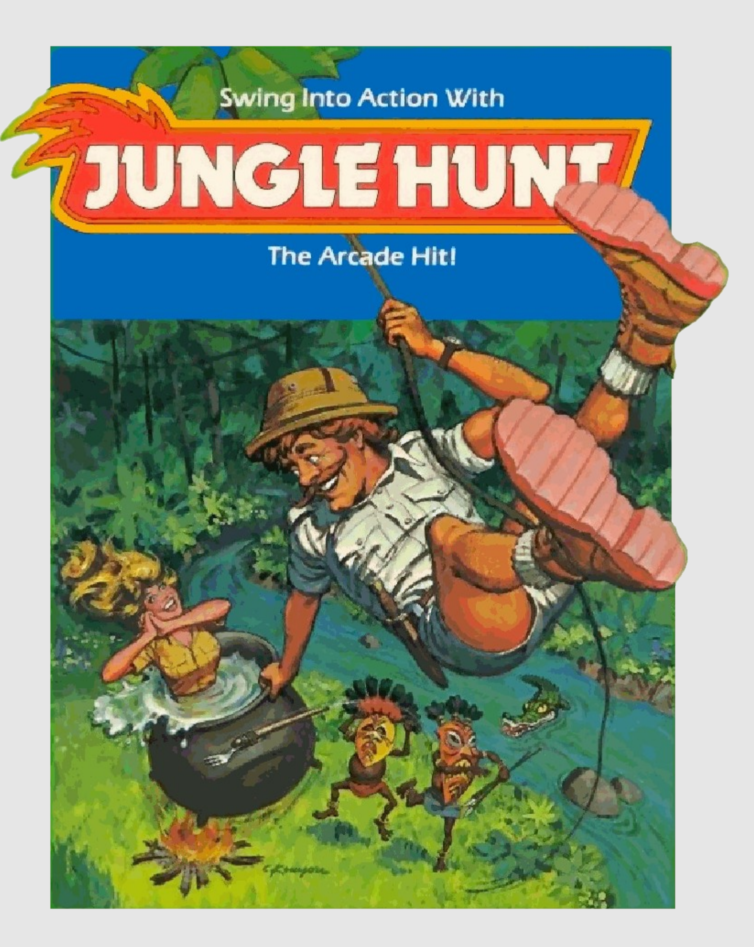
Caos Begins ha sido el ganador de la MSXDev del año pasado, pero no el juego más especial de dicha edición. Txupinazo! Es un bofetón de aire fresco, lo más original que hemos visto en mucho tiempo, y de lo más mimado en años en cuanto a detalles.

La música SCC, con un banjo perfectamente definido, es el punto clave en todo el conjunto, que junto a los divertidos gráficos hacen de este juego una experiencia única e irrepetible.

De momento sólo está disponible para emular, una pena:

http://msxdev.msxblue.com/index2.htm





Regreso a la jungla

Este juego nos trae buenos recuerdos, de la época en que Konami hacía juegos sencillos en plan arcade para MSX y los gráficos aprovechaban las capacidades del sistema para ofrecer un buen nivel. Pero no es Konami la autora.

El juego

Jungle Hunt nos pone en la piel de un explorador que debe salvar a la chica de turno de las pérfidas garras de los caníbales (los cuales ya tienen encendido el puchero y todo).

Originalmente hiba a ser Tarzán, de ahí esa primera fase con lianas, pero cosas del copyright desembocaron en el explorador que vamos a llevar. Como curiosidad, en el nivel del agua está el srpite original de Tarzán pero con gorro... al fin y al cabo nadie hiba a quejarse porque el explorador se quitara la ropa al tirarse al agua.

Gráficos

El acabado es exquisito, no sólo es fiel a la versión de Colecovisión, además la paleta de colores del MSX le sienta tremendamente bien.

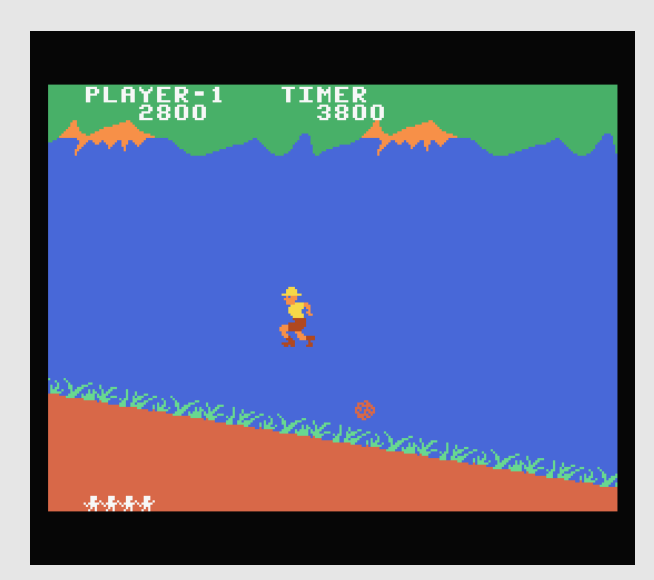
Por otra parte no era muy complicado, al fin la ROM original de Colecovison anda por debajo en cierta forma. ;)



El juego está genialmente convertido a MSX



La dificultad se adapta a todo tipo de jugadores



El scroll está muy conseguido y es bastante suave



La paleta de colores es muy agradable, mejorando todo

Sonido arcade 100%

Respecto al sonido, es totalmente fiel a la versión de Colecovision, si acaso con un tono más suave en MSX. No vemos sentido mejorar el sonido que ya tenía, es el que le pega a un juego así y mantiene el espíritu arcade.

Si acaso hubiera sido bueno crear una melodía para el juego.

Coleco y MSX matrimonio feliz

Desde que se descubriera como convertir juegos de Coleco a MSX este sistema está recibiendo multitud de conversiones que expanden aún más su catálogo.

Evidentemente tiene su trabajo, no es cosa de meter la ROM en un conversor y ya está, pero el resultado merece la pena si el juego es bueno.

El precio es casi un regalo, menos de 15 eurillos (12,5 euros con Matra), lo que no deja duda sobre su recomendación.

Gráficos	8
Audio	6
Juego	9

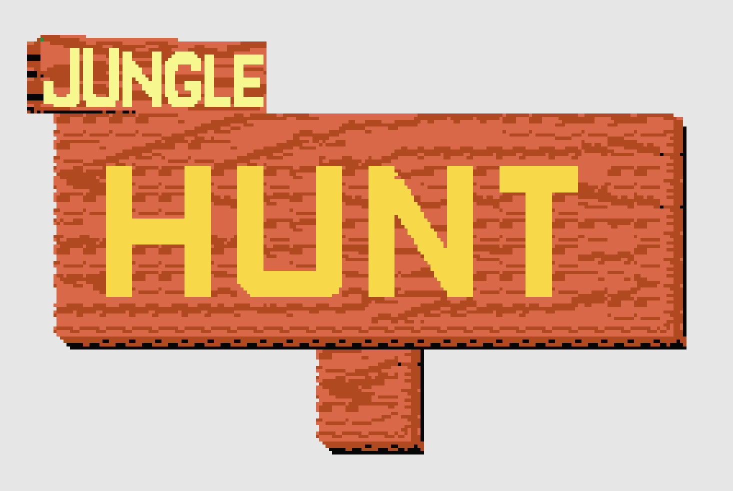
Nota 8

Tener un MSX y no comprar cartuchos como este es casi pecado.

Lo podeis adquirir por Internet en:

www.matranet.net
http://www.msx-universe.com/

También disponible en ferias y RU's.





Un regordete Indiana Jones

Realizar conversiones desde otras plataformas similares debe ser tomado en su justa medida. Al igual que Jungle Hunt era perfecto para MSX tal cual, no pasa lo mismo con otros títulos, aún cuando sean mejores como juego.

El juego

Ante todo hay que empezar con respeto, vamos a hablar de uno de los primeros juegos de plataformas y acción que combinaban la búsqueda de tesoros.

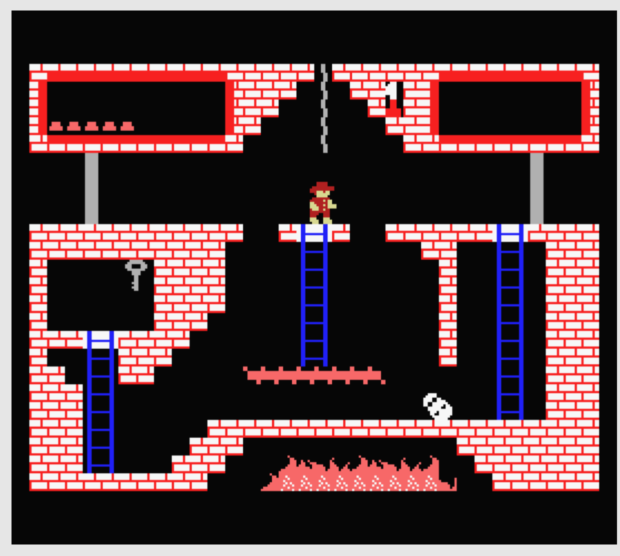
El objetivo es adentrarnos en un templo maya y conseguir el tesoro de Montezuma, para lo que vamos a tener que afrontar multitud de saltos, obstáculos, y enemigos que protegen la tumba debido a su maldición.

Gráficos

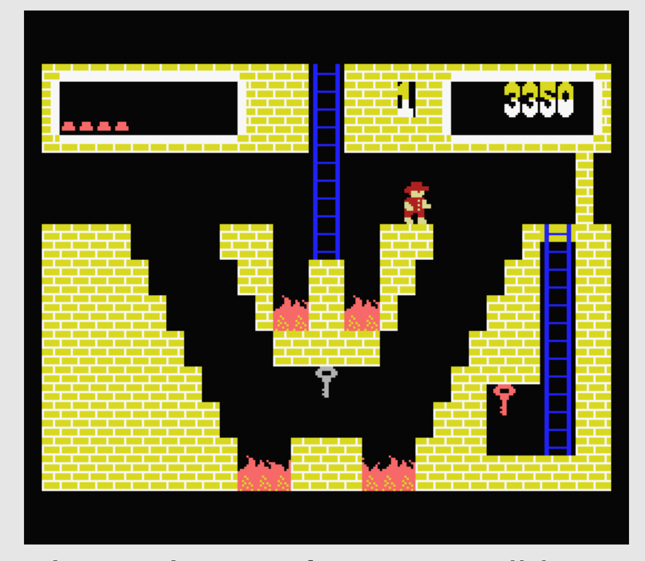
Llegamos al quid de la cuestión, ¿es posible mejorar el juego en MSX? La respuesta es un rotundo SI.

Uno ve la pantalla y sabe que se pueden meter muchos detalles, más color, esas cosillas que lo hubieran elevado a la categoría de obra maestra para un MSX.

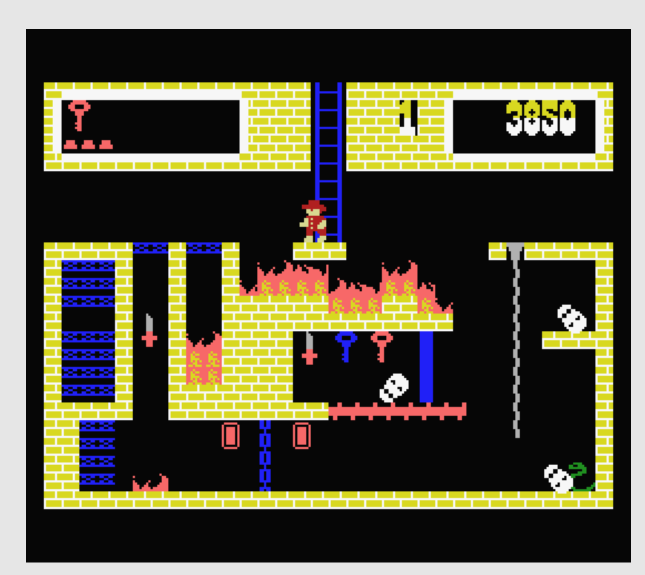
Por lo menos la pantalla de presentación está muy currada, y es el ejemplo de lo que pudo haber sido con un poco de esfuerzo.



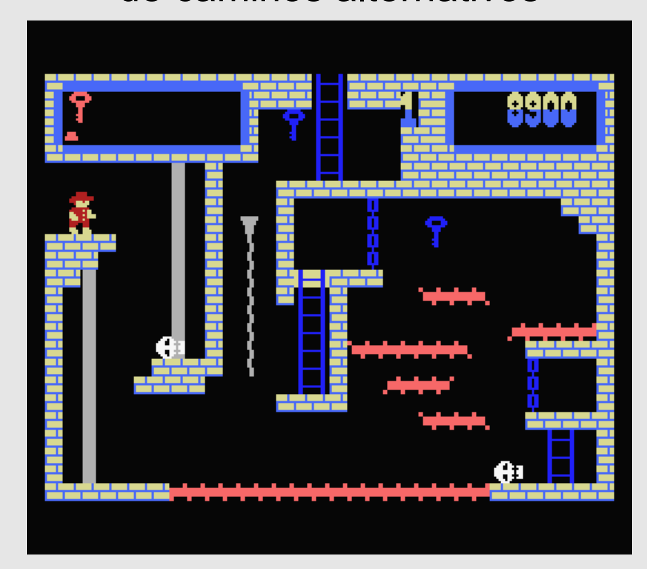
Esta pantalla de por si es todo un clásico de los videojuegos



Los saltos están muy medidos, puede llegar a desesperar



El mapeado es variado y lleno de caminos alternativos



La dificultad baja conforme le vayamos cogiendo el "punto"

Sonido, el justo

Olvidad melodías de ensueño, aquí sólo hay FX y muy primarios. Las pocas notas se oyen al coger objetos y no es para tirar cohetes. Otro punto muy mejorable.

Un gran juego deslucido

Este es el ejemplo de cómo se puede perder una oportunidad de oro para realizar una conversión.

Como juego es de 10, no encontraréis nada más entretenido, y rivaliza de tu a tu con grandes de MSX como Maze of Galious en lo que a diversión se refiere. Se trata de un imprescindible por el juego que es, pero pide a gritos un retoque en los sprites para ponerse al día. Necesita un lavado de cara.

El precio es el mismo que Junge Hunt, un gran chollo para los que tenemos un MSX real, y una oportunidad única para tener toda una leyenda de los videojuegos en nuestra colección.

Gráficos	6
Audio	5
Juego	10

Nota 7

Personalmente uno de mis juegos favoritos de todos los tiempos.

Lo podeis adquirir por Internet en:

www.matranet.net
http://www.msx-universe.com/

También disponible en ferias y RU's.



Entornos 3D en lenguaje BASIC (y II)

Motor gráfico FPS/RPG en primera persona para ZX Spectrum

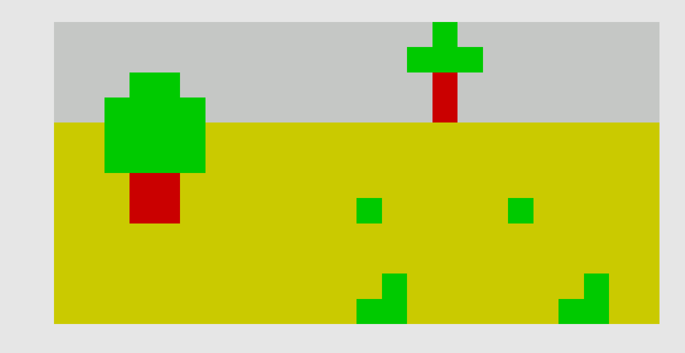
Parece increible que desde BASIC sea posible crear un juego con vista 3D, ¿verdad? Pues vamos a mejorarlo un poco más a costa de sacrificar un poco de velocidad.

Nueva perspectiva

Para empezar hay que mejorar la perspectiva, es decir que realmente veamos todo como si lo miráramos con nuestros ojos.

Hasta ahora hemos situado los objetos en tres alturas posibles según su distancia, pero siempre en la misma posición horizontal. Eso no es correcto, ya que cuanto más alejados estamos del objeto más objetos podemos ver en su misma línea de horizonte.

Es decir, unir los objetos según la distancia en la que se encuentren



Bien, esto ya es otra cosa, ahora sí tenemos una visión más realista del asunto.

Como podéis observar los objetos se van desplazando a los lados conforme se van acercando. Lo ideal sería que el la lejanía se pudieran ver más objetos (5 estaría bien), pero eso sería ya mucha carga para el BASIC y preferimos reservar la potencia para otros asuntos más interesantes.

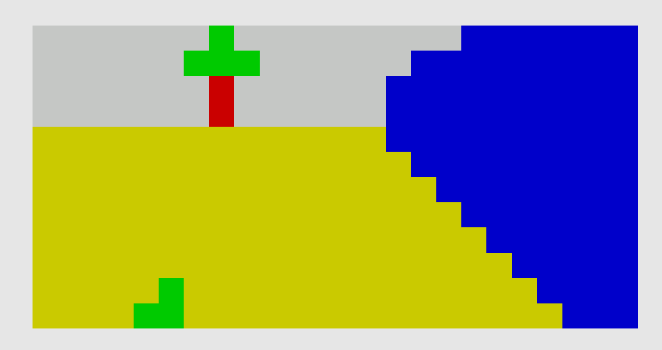
Con la nueva perspectiva también ganamos en una cosa: paredes.

Poniendo puertas al campo

Efectivamente, la nueva perspectiva nos permite realizar muros, paredes, y un sinfín más de cosas.

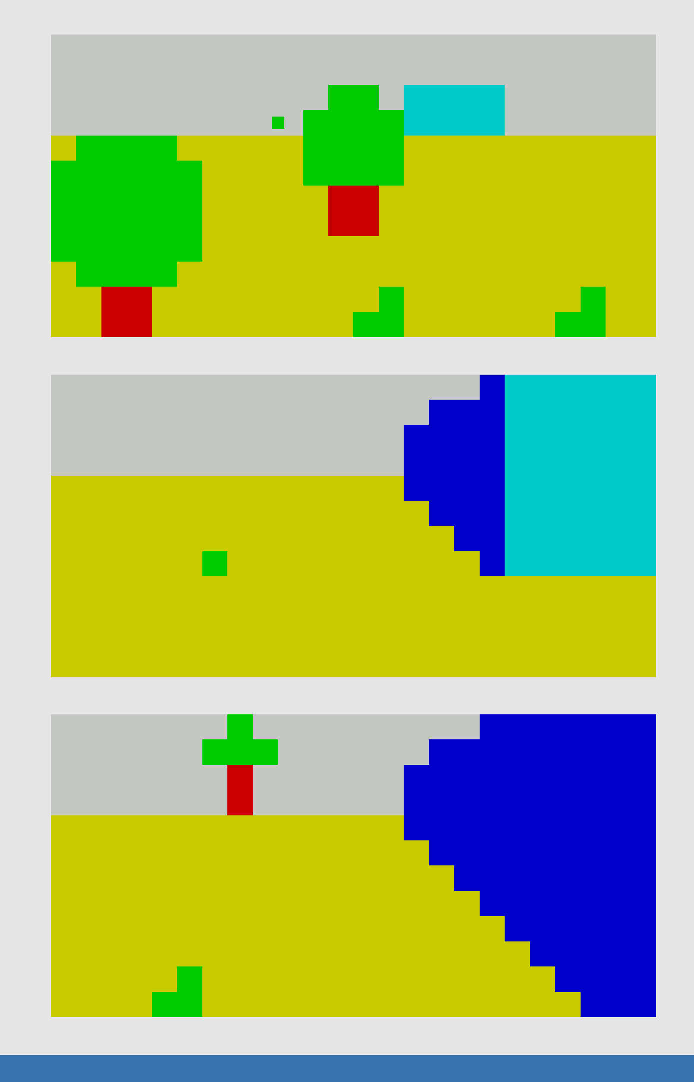
Basta con que dibujemos en un papel como sería un muro en perspectiva y lo dividamos horizontalmente en dos pedazos.

Es decir, un muro que ocupe todo el lateral de la pantalla se verá ocupando los dos objetos más cercanos de ese lado, y cada trozo representa una posición.



Ahora viene la pregunta... ¿dos trozos? ¿no son tres en los que se divide el muro?

Este es el truco:



Magia Potagia

Cuando dibujamos cada trozo de muro debemos dibujar el lateral y la parte frontal, es decir pensar en el muro como un cubo.

De esta forma los muros más alejados sólo podemos verlos frontalmente, ya que no podemos ver los laterales.

En la parte intermedia el lateral del muro y la parte frontal tapan el muro más alejado.

En la parte más cercana, el muro sólo puede verse lateralmente, y oculta la parte frontal de la parte de distancia media.

Eso en los laterales, los objetos que tenemos justo enfrente de la visión sólo podemos observarlos en su frontal, como un rectángulo, no hay laterales porque perdemos de vista la perspectiva de los mismos.

Pero no sólo tenemos este truco, hay otro más importante y es el orden.

Para que en cualquier posibilidad tengamos un bonito pasillo en 3D es necesario dibujar primero los objetos laterales, y dejar para el final los objetos del frente.

La secuencia de dibujo sería así:

- Horizonte más alejado: izquierda, derecha, frente

- Horizonte medio: izquierda, derecha, frente

- Horizonte más cercano: izquierda, derecha, frente

OJO: no debe haber muro justo en la posición frontal más cercana, ya que es la posición que ocupamos como observador (sería absurdo ver sólo una pared de muro en la pantalla).

Como anécdota podemos decir que en el horizonte más lejano dará igual el orden de dibujo, ya que los muros sólo vamos a verlos de forma frontal.

Ahora vamos a ver cómo queda todo esto en el código que tenemos realizado hasta ahora:

```
120 REM rutina movimiento
130 LET s$=INKEY$
140 IF s$="o" AND x>3 AND a(x-1,y)<5
THEN LET x=x-1: GO TO 1000
150 IF s$="p" AND x<17 AND a(x+1,y)<5
THEN LET x=x+1: GO TO 1000
160 IF s$="q" AND y>3 AND a(x,y-1)<5
THEN LET y=y-1: GO TO 1000
170 IF s$="a" AND y<17 AND a(x,y+1)<5
THEN LET y=y+1: GO TO 1000
180 GO TO 130
```

Como podemos observar, el primer cambio realizado es detectar si en el punto de vista del observador hay un objeto que no se puede traspasar (ej. una pared). Para ello, en este caso, hemos decidido que todo valor mayor de 5 en la matriz es un objeto que no podemos traspasar.

Esta chorrada ya nos permite movernos por un mapa inteligente: rios, muros de vado, casas, etc

Ahora veamos la forma de dibujar.

```
2030 GO TO (2040-(a(x-1,y-1)))
2035 PRINT AT 8,10...
2037 GO TO 2040
2038 PRINT AT 11,13...
2039 PRINT AT 9,11...
```

Hemos recortado las líneas para dejar sólo lo que nos interesa explicar.

Como podeis ver estamos dibujando primero el lateral izquierod, luego el derecho, y en último lugar la parte central.

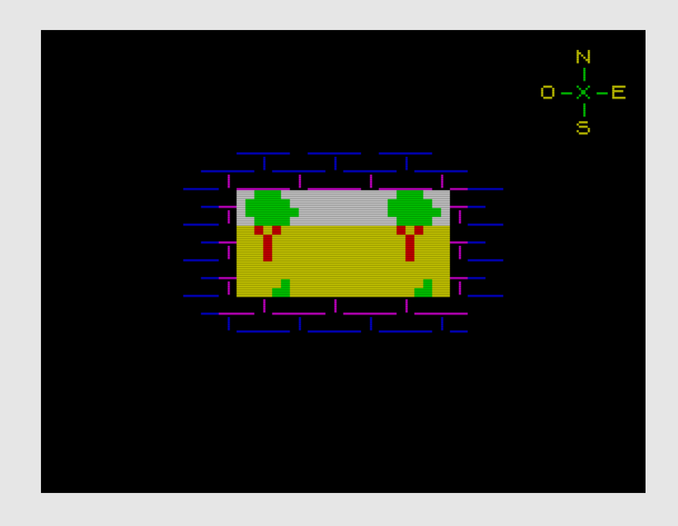
El motivo ya lo sabeis, que los trozos de muro/pared/etc se tapen entre ellos y den un aspecto realista.

Y esto es todo el secreto para dibujar en 3D, no hay más, ahora viene lo complicado: cambiar el punto de vista para poder girar y cambiar la orientación del punto de vista.

Es hora de poder movernos como en un verdadero RPG en primera persona, pero vamos a tener un costo: todo irá mucho más lento, pero siendo un RPG no pasará nada.

Hay que ser “profezioná”

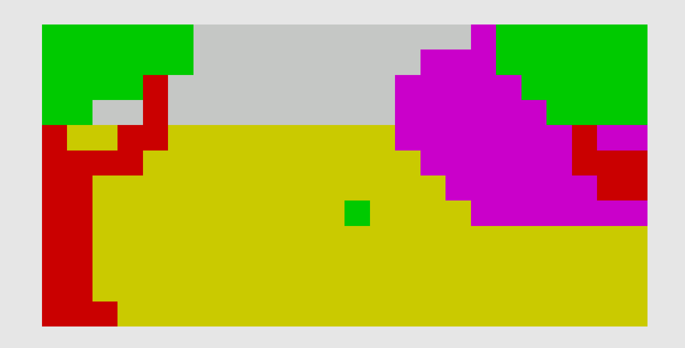
Para empezar vamos a dar un lavado de cara a la pantalla de juego, se acabó la chapuza de un rectángulo y nada más, unos ladrillos y un compás es algo mucho mejor para alegrar la vista.



Esto ya es otra cosa, con un “decorado” en pantalla parece hasta otro juego.

Otro avance que vamos a poner es usar unos árboles más currados, usando los mismos bloques, además de usar un color de fondo para los muros, en vez de bloques.

Lo de usar un color de fondo para los muros responde a la necesidad que cuando pongamos algo por delante (ej, un árbol) el color de fondo transparente nos permita que no se quede feo.



Como veis el resultado es mucho más agradable y realista, los árboles ya no parecen de juguete y se superponen a los muros sin perder el color de los mismos.

Pero dejemos de lado los cambios estéticos y entremos en el código.

Para empezar inicializaremos las variables, la matriz, y meteremos una variable nueva que defina la orientación de avance.

```
20 FOR b=1 TO 20: FOR c=1 TO 20: LET d=INT (RND*4): LET a(b,c)=d: NEXT c: NEXT b
100 REM inicializamos variables
110 LET x=10: LET y=10: LET p=1: GO TO 1000
```

Seguiremos usando una matriz de 20x20 para el mapa, el cual vamos a rellenar de escenario, y metemos la variable “p” que nos indica:

p=1 mira hacia el norte
p=2 mira hacia el este
p=3 mira hacia el sur
p=4 mira hacia el oeste

Ahora tenemos que modificar el motor para que incluya la orientación, para lo cual vamos a tener que hacer dos cosas: que el movimiento detecte las colisiones según la orientación, y que la pantalla se dibuje según la orientación también.

Para ellos vamos a usar dos cosas: tablas de saltos para el movimiento (como en el dibujado) y calcular las tablas de saltos del dibujo previamente según la orientación que tengamos.

Movimiento

Como es lógico necesitamos una teclas más acorde a los giros, así que vamos a usar AWSD para movernos y QE para girar, teclas muy similares a las usadas en juegos de PC.

Veamos el código para la tecla “a”:

```
120 REM rutina movimiento
130 LET s$=INKEY$
140 IF s$="a" THEN GO TO 141+p
141 GO TO 150
142 LET x=x-(a(x-1,y)<5): GO TO 1000
143 LET y=y-(a(x,y-1)<5): GO TO 1000
144 LET x=x+(a(x+1,y)<5): GO TO 1000
145 LET y=y+(a(x,y+1)<5): GO TO 1000
```

Bien, como podéis ver la diferencia estriba en que antes de realizar un movimiento a izquierda debemos verificar si hay un obstáculo según la posición de giro que tenemos.

Para ellos hemso usado una tabla de saltos en función del giro (p).

El resto de teclas de movimiento es:

```
150 IF s$="d" THEN GO TO 151+p
151 GO TO 160
152 LET x=x+(a(x+1,y)<5): GO TO 1000
153 LET y=y+(a(x,y+1)<5): GO TO 1000
154 LET x=x-(a(x-1,y)<5): GO TO 1000
155 LET y=y-(a(x,y-1)<5): GO TO 1000
160 IF s$="w" THEN GO TO 161+p
161 GO TO 170
162 LET y=y-(a(x,y-1)<5): GO TO 1000
163 LET x=x+(a(x+1,y)<5): GO TO 1000
164 LET y=y+(a(x,y+1)<5): GO TO 1000
165 LET x=x-(a(x-1,y)<5): GO TO 1000
170 IF s$="s" THEN GO TO 171+p
171 GO TO 180
172 LET y=y+(a(x,y+1)<5): GO TO 1000
173 LET x=x-(a(x-1,y)<5): GO TO 1000
174 LET y=y-(a(x,y-1)<5): GO TO 1000
175 LET x=x+(a(x+1,y)<5): GO TO 1000
```

No hay secretos, simplemente variamos las coordenadas a revisar según el valor de “p”. Si no lo entendeis es más sencillo seguir los ajustes si escribis una cuadrícula en un papel.

La posición central es x,y, la de la izquierda mirando al norte x-1, derecha x+1,y, etc. Es muy sencillo.

Para realizar el giro tampoco es que haya que ser un lumbreras, pensad en como gira la manecilla de un reloj:

```
180 IF s$="q" THEN LET p=p-1: GO TO
1000
190 IF s$="e" THEN LET p=p+1: GO TO
1000
200 GO TO 130
```

Ya hemos terminado la rutina de movimiento, ahora viene lo complicado. La rutina de visión comenzará comprobando si nos salimos del mapa:

```
1000 REM rutina de vision
1001 IF x<3 THEN LET x=3: GO TO 1090
1002 IF y<3 THEN LET y=3: GO TO 1090
1003 IF x>17 THEN LET x=17: GO TO
1090
1004 IF y>17 THEN LET y=17: GO TO
1090
```

Y ahora comprobamos si el giro ha excedido los valores de p, corrigiéndolo si es así (p<0 o p>4):

```
1090 GO TO (1093+p)
1093 LET p=4: GO TO 1100
1097 GO TO 1100
1098 LET p=1
```

Por último lugar tenemos la parte más coñazo, y es que vamos a crear 9 variables que nos indiquen las posiciones a mirar dentro de la matriz del mapa según el giro que tengamos, otra tabla de saltos más:

```
1100 GO TO (1100+p)
1101 PRINT AT 0,29; INK 6;"N";AT
1,29; INK 4;"|"; INK 6;AT 2,27;"O"; INK
4;"-X-"; INK 6;"E";AT 3,29; INK 4;"|";AT
4,29; INK 6;"S":
LET pa=(2010-(a(x-1,y-2))):
LET pb=(2020-(a(x,y-2))):
LET pc=(2030-(a(x+1,y-2))):
LET pd=(2040-(a(x-1,y-1))):
LET pe=(2050-(a(x+1,y-1))):
LET pf=(2060-(a(x,y-1))):
LET pg=(2070-(a(x-1,y))):
LET ph=(2080-(a(x+1,y))):
LET pj=(2090-(a(x,y))): GO TO 1200
```

Las 9 variables son, de izquierda a derecha y de atrás a delante:

pa	pb	pc
pd	pf	pe
pg	pj	ph

Puede parecer un lío, pero si usais la cuadrícula de antes es simple.

Creo que ya no hace falta explicaros más, lo demás es bastante sencillo, y en los ficheros del curso teneis el código para poder mirarlo y ver el resto de líneas.

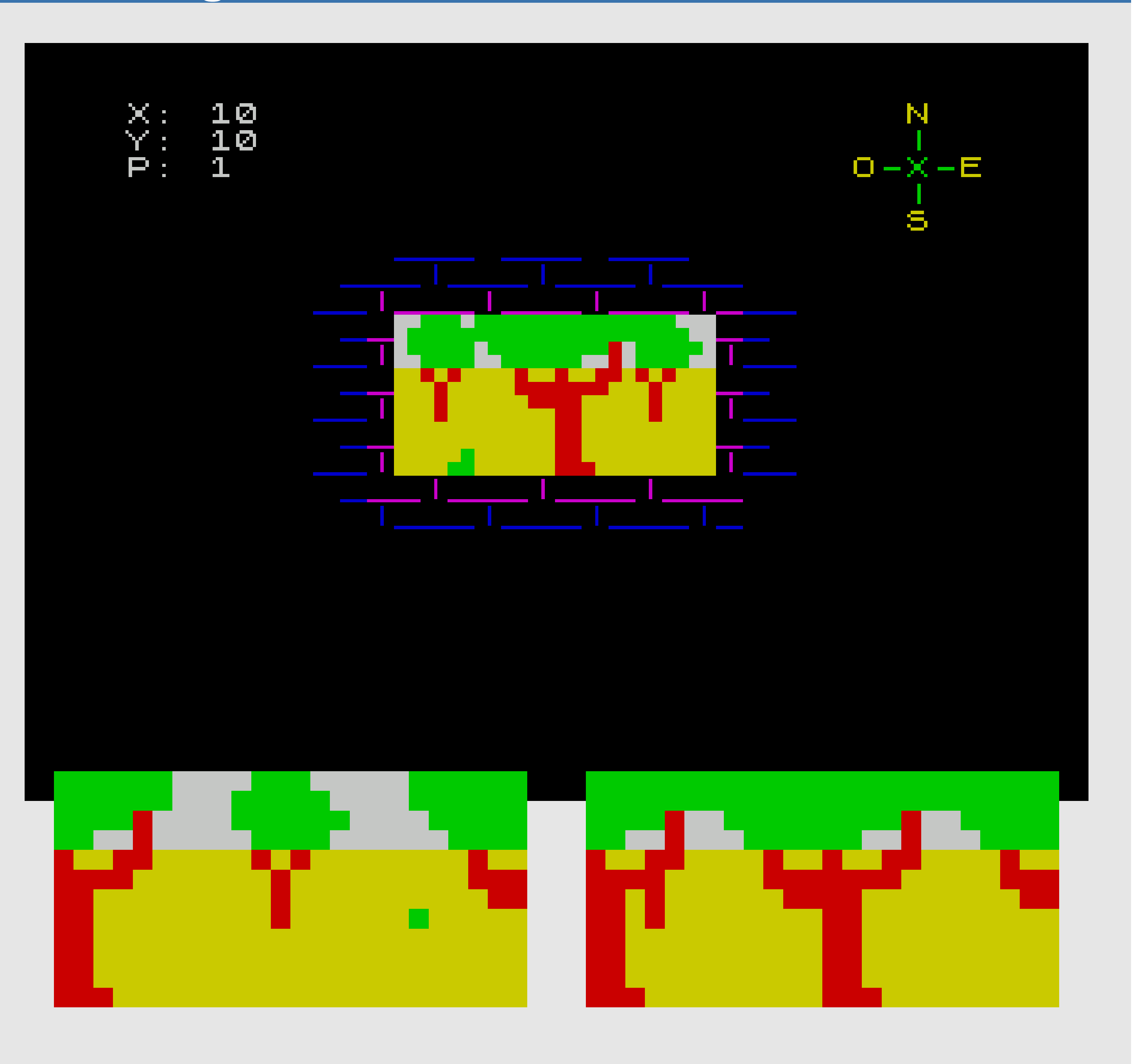
Vereis que cuando llegueis a la parte del dibujado se debe emplear un goto pa, pb, etc en vez del método clásico empleado hasta ahora.

Y nada más, espero que este curso os haya servido para ver que es posible crear otro tipo de juegos en lenguaje BASIC inéditos hasta ahora.

Empleando giro podemos crear un estupendo RPG, y usando el primer sistema es posible crear algo parecido a un FPS... o, porqué no, incluso un juego de coches.

Si sólo usamos el motor gráfico es muy sencillo crearnos una matriz de 6x100, por ejemplo, y meter un circuito de rallies en el que sólo avanzamos pero podemos movernos de lado a lado para evitar obstáculos.

http://www.bytemaniacos.com/ficheros



Sprites desde ensamblador (I)

Generación de gráficos avanzados para ZX Spectrum

La pantalla del ZX Spectrum es la mayor desgracia para el programador, ya que su estructura no es lineal y las rutinas por lo tanto no son tan simples como en otros sistemas.

Y es que los ingenieros de Sinclair pensaron en ella desde su punto de vista, no como programadores, lo que puede llevar a auténticos dolores de cabeza si no se tienen las cosas claras desde el principio.

Así que comencemos desde lo más simple, tamaño, posición, y estructura del puzzle que representa la pantalla en un ZX Spectrum.

Debemos tener una cosa clara desde el principio: la pantalla es una mezcla de dos pantallas realmente, y cada una es totalmente diferente de la otra.

¿Dos pantallas?

Si, efectivamente, realmente tenemos dos pantallas que mezcladas nos dan la que vemos:

Pantalla de pixels
Pantalla de atributos

La primera son los puntos, nada más, y la segunda da color a la primera. Para muchos esto no es así, prefieren pensar que es sólo una, pero todo queda más claro si lo vemos desde este punto de vista.

En primer lugar vamos a comentar la más sencilla de las dos, la de atributos, y después pasaremos a la que realmente nos traerá de cabeza.

Comencemos con los colorines.

Pantalla de atributos

La pantalla de atributos es así:

- Resolución de 32x24 cuadros
- Cada cuadro es un byte de color
- Comienza en 22528 (decimal)
- Tamaño total 768 bytes

Se trata de una pantalla muy simple, comienza en la esquina superior izquierda y cuando llegamos al final de la fila pasa a la siguiente linealmente.

Podemos crear cosas con esta pantalla sin emplear ni un sólo pixel, sólo modificará el color de cada cuadro (8x8 pixels de tamaño).

La correspondencia de colores es la siguiente:

TINTA

NEGRO	0	00000000
AZUL	1	00000001
ROJO	2	00000010
MAGENTA	3	00000011
VERDE	4	00000100
AZULADO	5	00000101
AMARILLO	6	00000110
BLANCO	7	00000111

PAPEL

NEGRO	0	00000000
AZUL	8	00001000
ROJO	16	00010000
MAGENTA	24	00011000
VERDE	32	00100000
AZULADO	40	00101000
AMARILLO	48	00110000
BLANCO	56	00111000

Cada byte de color se divide en dos: papel y tinta. Los tres bits de menor peso definen la tinta y los tres siguientes el color del papel.

El papel es visible siempre, pero la tinta sólo la veremos si hay pixels.

Recordemos que la pantalla final es la mezcla de los colores de atributo con la pantalla de pixels, por lo que si no usamos pixels sólo podemos usar los atributos de papel para obtener algo en pantalla.

Puede parecer una chorrada, pero sólo con modificar los atributos de papel se pueden hacer auténticas vilguerias.

Para obtener el valor de atributo basta con sumar los valores de tinta y papel. Por ejemplo:

TINTA ROJA: 2
PAPEL AMARILLO: 48

VALOR DE ATRIBUTO: 50

La pantalla de atributos comienza en 22528 y finaliza en 23295 (768 bytes). Un borrado de esta pantalla sería:

```
ld hl, 22528
ld de, 22529
ld bc, 767
ld (hl),a
ldir
```

Donde el acumulador es el valor de atributo para el borrado.

Triquiñuelas

Antes de pasar a explicar el mapa de pantalla vamos a indicar algo muy útil de cara a realizar nuestras rutinas, y es cómo acceder a cada pantalla de la forma más sencilla posible.

Una dirección de memoria se indica con dos bytes, lo que nos permite acceder a 64K bytes.

Si olvidamos el byte bajo, resulta que en el byte alto pasa algo curioso si tenemos el byte bajo a 0:

Si sólo activamos el bit 6 (64 en decimal) apuntamos al comienzo del área de pantalla (zona de pixels), ya que la dirección sería 16384.

Si activamos los bits 6, 4, y 3 (88 en decimal) apuntamos al comienzo del área de atributos, ya que la dirección sería 22528.

Pantalla de pixels

La pantalla de pixels es algo abstracta debido a su estructura, vamos poco a poco:

Comienza en 16384 (64+0 en bytes)
Resolución: 256x192
Cada byte son 8 pixels
Longitud total:6144

Una línea de pantalla tiene 32 bytes, cada uno de los cuales representa un pixel por bit.

Pero la memoria no es lineal, se divide en tercios y cada tercio está entrelazado. Esto quiere decir que:

- Cada 2K bytes cambiamos de tercio de pantalla (sumando 8 al byte alto de dirección)

- Si queremos saltar a la siguiente línea tenemos que sumar 8x32 (256 bytes), ya que la siguiente línea no es contigua.

¿Complicado? Veamos un ejemplo.

Para empezar por lo simple vamos a imprimir un trozo de muro:

```
        ld de, 16384        ; 16384 es la posición donde vamos a dibujar
        ld hl, muro         ; muro es el sprite de dibujar de 8x8 pixels
        call print          ; print será nuestra rutina de sprites
        ret

; rutina de impresión de un sprite de 8x8 pixels sin atributos

print   ld b, 8             ; ponemos en b el nº de líneas del sprite
draw    ld a,(hl)           ; hl indica la posición del sprite en memoria
        ld (de),a           ; de indica la posición de pantalla
        inc hl              ; pasamos a la siguiente línea de sprite
        inc d               ; pasamos a la siguiente línea de pantalla
        djnz draw           ; decrementa B y si es cero deja de saltar a draw
        ret

; sprite de un trozo de muro

muro defb 223,223,223,0,253,253,253,0
```

Hemos supuesto varias cosas:

- Que vamos a usar posiciones absolutas de carácter
- Que el sprite es de 8x8 pixels sin atributos

De esta forma basta con escribir en el byte de pantalla que hemos indicado en de (en este caso el primero de toda la pantalla) e ir incrementando el valor del byte alto de dirección (d) para saltar a la línea contigua (estamos sumando 256 con este truco en cada incremento de d).

Evidentemente esta rutina es una chapuza, ya que debemos darle nosotros directamente la dirección de memoria que se corresponda con la primera línea del carácter donde queremos poner el sprite. La solución es realizar una rutina que primero nos calcule el lugar donde situar el sprite en coordenadas x, y de toda la vida.

```
        ld d, 12        ; D será la posición vertical del sprite en caracteres
        ld e, 10        ; E será la posición horizontal del sprite en caracteres
        ld hl, muro
        call print
        ret

print   ld a, d         ; cargamos el valor vertical
        and 7           ; nos quedamos con la posición en el tercio (dejamos los tres primeros bits)
        rrca            ; rotamos para dejar su valor en múltiplos de 32 (linea)
        rrca
        rrca
        and 224         ; borramos el resto de bits por si las moscas
        or e            ; sumamos el valor horizontal
        ld e, a         ; E preparado
        ld a, d         ; cargamos el valor vertical
        and 24          ; borramos todo salvo los bits 3 y 4 que indican el tercio de pantalla
        or 64           ; nos posicionamos a partir de 16384 (16384=64+0 en dos bytes)
        ld d, a         ; D preparado
        ld b, 8         ; ponemos en b el nº de líneas del sprite
draw    ld a,(hl)       ; hl indica la posición del sprite en memoria
        ld (de),a       ; de indica la posición de pantalla
        inc hl          ; pasamos a la siguiente línea de sprite
        inc d           ; pasamos a la siguiente línea de pantalla
        djnz draw       ; decrementa B y si es cero deja de saltar a draw
        ret
muro defb 223,223,223,0,253,253,253,0
```

Como veis, la creación de las rutinas está íntimamente ligada a saber lo que sucede a nivel de bit con los bytes de direccionamiento.

Cada tercio de pantalla se divide en 8 caracteres de alto, es decir, si quitamos todos los bits con un AND 7 borraremos todos los bits salvo los tres que nos indican la posición en el tercio.

Si por el contrario borramos todos los bits salvo el tercero y el cuarto (AND 24) obtenemos el tercio de pantalla en el que vamos a movernos.

Osea, el byte de valor vertical representado en bits es:

b7 b6 b5 b4 b3 b2 b1 b0

b2, b1, y b0 es la posición en el tercio

b4 y b3 es el tercio de pantalla

Pasamos a ver como dibujar un sprite de 16x16 pixels en monocromo.

```
print   push de         ; salvamos la posición del sprite en x,y
        call cdrw       ; calculamos posición en memoria
        ld b, 8         ; preparamos B para 8 pixels de altura
        call draw       ; dibujamos una fila del sprite de 8x16
        pop de          ; recuperamos la posición del sprite en x,y
        inc d           ; incrementamos una línea en el sprite
        call cdrw       ; calculamos posición en memoria
        ld b, 8         ; prepramos B para 8 pixels de altura
        call draw       ; dibujamos la segunda fila del sprite 8x16
        ret
draw    ld a,(hl)       ; recogemos el byte izquierdo del sprite
        ld (de),a       ; lo imprimimos
        inc hl          ; pasamos al siguiente byte del sprite
        inc e           ; pasamos al siguiente byte de pantalla
        ld a,(hl)       ; recogemos el byte derecho del sprite
        ld (de),a       ; lo imprimimos
        inc hl          ; pasamos al siguiente byte del sprite
        dec e           ; volvemos atrás horizontalmente
        inc d           ; pasamos a la siguiente línea de pantalla
        djnz draw       ; decrementa B y si es cero deja de saltar a draw
        ret
cdrw    ld a, d         ; recuperamos el valor vertical
        and 7           ; nos quedamos con la posición en el tercio
        rrca            ; rotamos para dejar su valor en múltiplos de 32 (linea)
        rrca
        rrca
        and 224         ; borramos el resto de bits por si las moscas
        or e            ; sumamos el valor horizontal
        ld e, a         ; E preparado
        ld a, d         ; cargamos el valor vertical
        and 24          ; borramos todo salvo los bits que indican el tercio de pantalla
        or 64           ; nos posicionamos a partir de 16384
        ld d, a         ; D preparado
        ret
```

Hemos supuesto que el sprite se compone de 16x16 pixels, o lo que es lo mismo, 32 bytes de tamaño.

El sprite debemos crearlo con orden lineal, sin entrelazar.

Ya sólo nos falta saber cómo aplicar el color al sprite.

Coloreando

Para colorear vamos a añadir 4 bytes más al sprite, que serán los valores de atributo correspondientes.

Primero debemos realizar el cálculo de la posición del color en la pantalla de atributos:

```
ld a,d        ; cargamos el valor vertical
rra           ; multiplicamos por 32 (tamaño de línea)
rra
rra
and 3         ; nos quedamos con los dos bits bajos
or 88         ; apuntamos a la memoria de atributos
ld d,a        ; ya tenemos d listo
```

Como podemos ver, al ser lineal la pantalla de atributos, no tenemos que modificar el valor del registro E. El valor de D se obtiene multiplicando por el número de caracteres que tiene una línea

Para imprimir los colores, basta:

```
ld a,(hl)     ; color izquierdo de la línea de sprite
ld (de),a     ; ponemos el color en la pantalla de atributos
inc e         ; pasamos al siguiente caracter
inc hl        ; pasamos al siguiente color del sprite
ld a,(hl)     ; color derecho de la línea de sprite
ld (de),a     ; ponemos el color en la pantalla de atributos
dec e         ; retrocedemos un carácter para la siguiente línea
inc hl        ; pasamos al siguiente color
ret
```

Esta rutina la tenemos que repetir para cada línea de caracteres que ocupe el sprite.

Y de momento nada más, para el próximo capítulo vamos a introducir el concepto de máscara y aprenderemos a crear bloques de sprites para los mapeados.

Es importante, no obstante, que adapteis las rutinas de sprites a vuestras necesidades, ya que la velocidad varia mucho si usamos una rutina adaptada o no. Por ejemplo, la rutina para un sprite de 8x8 en monocromo es muchísimo más rápida que la de 16x16 en color, pero cada una tiene sus ventajas.

Podeis ir viendo código fuente, preparado para PASMO en:

http://www.bytemaniacos.com/ficheros/curso_sprites/

El espacio

Para muchos el juego espacial por excelencia fué Elite, una obra maestra que permitió que nos perdiéramos por la galaxia mercadeando y luchando. Pero hubo un clon, Starfox, que hizo las delicias de los amantes de la acción sin complicaciones. Con sus estaciones espaciales, sus piratillas, etc, resultaba la mar de entretenido sin tantas opciones y teclas como Elite.

